NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Introduction to Computational Thinking

Mini Project

Real-time Canteen Information System: FOOGLE

Group Member:

Carissa Lee Xue Wei	U1921455G
Au Yi Xian	U1923991D
Bachhas Nikita	U1921630F

Lab Group Number: FE1

Date of Submission: 13 November 2019

Contents

Problem Statement

In Nanyang Technological University (NTU), we are often faced with the challenge of dealing with large crowds due to the student population of 32, 015 [1], excluding staff and faculty members. This situation is felt particularly at our eating hub, such as the North Spine Canteen. Being a popular canteen, people often have to deal with long queues and unknown waiting times, especially during peak hours.

There are also times where people walk to the canteen only to realise that the particular stall you want is closed. This caused frustrating for everybody.

Solution

Our solution is to create a Real-Time Canteen Information Systems, named ‘FOOGLE’. Using both Tkinter and Python language, people experience more convenience when they wish to gain information about the North Spine Canteen.

Upon entering the program, users can first choose to either input their desired time and date of visit or use the current time and date.

Following this, they can gain information regarding the canteen and each stall. Such information will include operation hours and menus. Special menus include different meals available at different timings of the day.

Lastly, the app will also calculate and display the estimated waiting time for the user by prompting them to input the number of people that are currently queueing in front of him/her.

Store Information

Stall 1: Chicken Rice Waiting Time Per Pax: 1 min	Operating Hours: Monday - Friday: 0800 - 2030 Saturday: 0800 - 1430 Sunday & Public Holiday: OFF	
Menu:	Price:	Remark(s):
Steamed/Roasted Chicken Rice Set	$4.50	
Steamed/Roasted Chicken Rice	$3.00	
Lemon Chicken Rice	$3.00	
Curry Chicken Noodle	$3.00	

Stall 2: Hand-made Noodles Waiting Time Per Pax: 2 min	Operating Hours: Monday - Friday: 0800 - 2030 Saturday: 0800 - 1430 Sunday & Public Holiday: OFF	
Menu:	Price:	Remark(s):
Ban Mian	$3.00	
Mee Hoon Kway	$3.00	
U-Mian	$3.00	
Dumpling Ban Mian	$3.50	

Stall 3: Cantonese Roast Duck Waiting Time Per Pax: 1 min	Operating Hours: Monday - Friday: 0800 - 2030 Saturday: 0800 - 1430 Sunday & Public Holiday: OFF	
Menu:	Price:	Remark(s):
Roasted Duck Rice	$3.30	
Char Siew Rice	$2.80	
Roasted Meat Rice	$2.80	
Char Siew Noodle	$3.30	

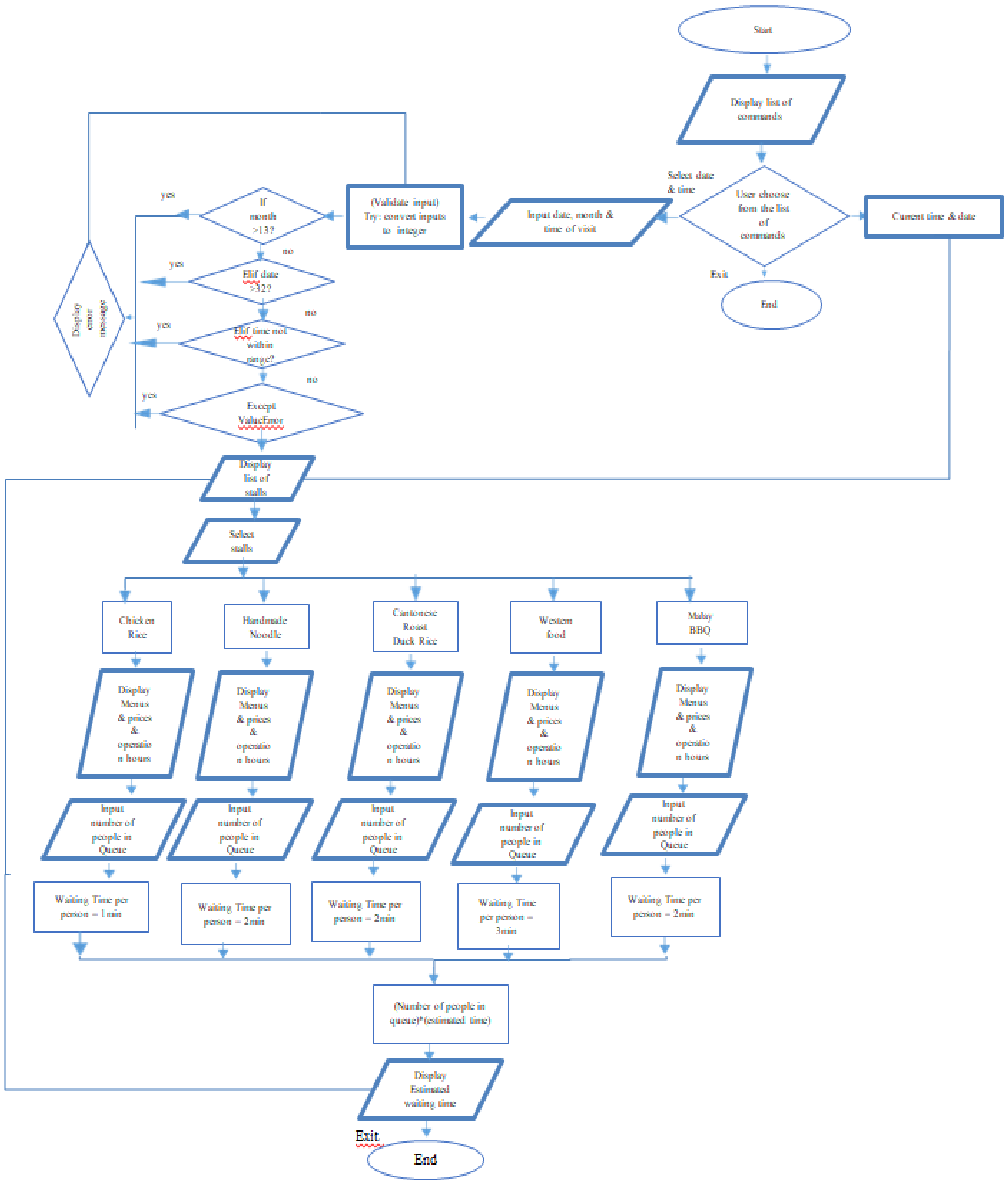
Stall 4: Western Food Waiting Time Per Pax: 3 min	Operating Hours: Monday - Friday: 0800 - 2030 Saturday: 0800 - 1430 Sunday & Public Holiday: OFF	
Menu:	Price:	Remark(s):
American Breakfast Set	$4.30	Breakfast Set: 0800 - 1130
Chicken Cutlet	$4.80	
Chicken Chop	$4.80	
Fish & Chips	$4.80	
Beef Steak	$6.30	

Stall 5: Malay BBQ Waiting Time Per Pax: 1 min	Operating Hours: Monday - Friday: 0800 - 2030 Saturday - Sunday & Public Holiday: OFF	
Menu:	Price:	Remark(s):
Lantong	$2.80	Breakfast Set: 0800 - 1130
Mee Rebus	$2.80	Breakfast Set: 0800 - 1130
Mee Siam	$2.80	Breakfast Set: 0800 - 1130
Mee Soto	$2.80	Breakfast Set: 0800 - 1130
Ayam Penyet	$4.00	
Fish Fillet	$4.00	

Algorithm Design

Flowchart

The flowchart is created to give a brief understanding of the key information needed to create the system. It also ensures a systematic flow of the system which can be seen in figure 1.



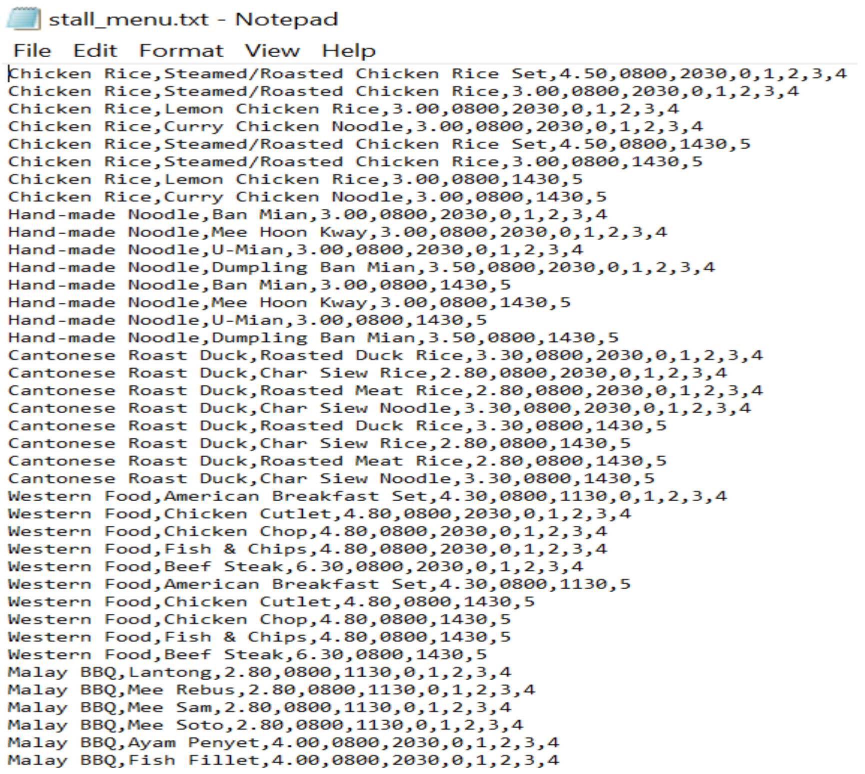
Start
Display list of commands
User choose from the list of commands
Select date & time
Current time & date
Exit
End
Input date, month & time of visit
(Validate input) Try: convert inputs to integer
If month >13?
yes
no
Elif date >32?
yes
no
Elif time not within range?
yes
no
Except ValueError
yes
Display error message
Display list of stalls
Select stalls
Chicken Rice
Handmade Noodle
Cantonese Roast Duck Rice
Western food
Malay BBQ
Display Menus & prices & operation hours
Input number of people in Queue
Waiting Time per person = 1min
Waiting Time per person = 2min
Waiting Time per person = 2min
Waiting Time per person = 3min
Waiting Time per person = 2min
(Number of people in queue)*(estimated time)
Display Estimated waiting time
Exit
End

Figure 1: Flow Chart

Brief Description of the Important User Defined Function

1. Store and display stall information/ menu

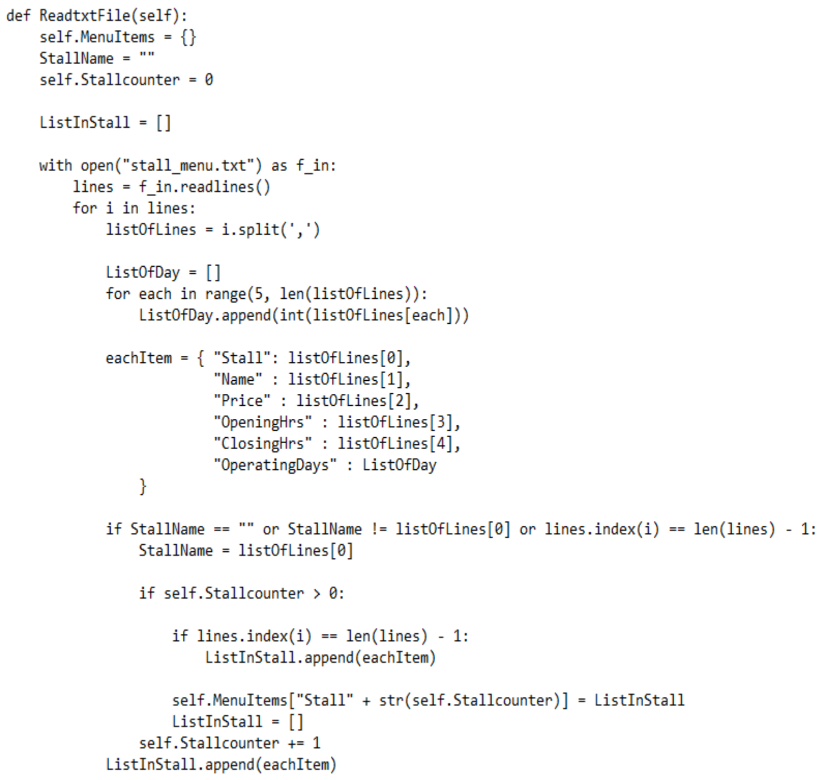
A text file is used to store the canteen stalls' details in Figure 2.
Every line: Stall Name, Menu, Price, Opening Time, Closing Time, Days
Days: 0 - Monday, 1 - Tuesday, 2 - Wednesday, 3 - Thursday, 4 - Friday, 5 - Saturday, 6 - Sunday



```
Chicken Rice,Steamed/Roasted Chicken Rice Set,4.50,0800,2030,0,1,2,3,4
Chicken Rice,Steamed/Roasted Chicken Rice,3.00,0800,2030,0,1,2,3,4
Chicken Rice,Lemon Chicken Rice,3.00,0800,2030,0,1,2,3,4
Chicken Rice,Curry Chicken Noodle,3.00,0800,2030,0,1,2,3,4
Chicken Rice,Steamed/Roasted Chicken Rice Set,4.50,0800,1430,5
Chicken Rice,Steamed/Roasted Chicken Rice,3.00,0800,1430,5
Chicken Rice,Lemon Chicken Rice,3.00,0800,1430,5
Chicken Rice,Curry Chicken Noodle,3.00,0800,1430,5
Hand-made Noodle,Ban Mian,3.00,0800,2030,0,1,2,3,4
Hand-made Noodle,Mee Hoon Kway,3.00,0800,2030,0,1,2,3,4
Hand-made Noodle,U-Mian,3.00,0800,2030,0,1,2,3,4
Hand-made Noodle,Dumpling Ban Mian,3.50,0800,2030,0,1,2,3,4
Hand-made Noodle,Ban Mian,3.00,0800,1430,5
Hand-made Noodle,Mee Hoon Kway,3.00,0800,1430,5
Hand-made Noodle,U-Mian,3.00,0800,1430,5
Hand-made Noodle,Dumpling Ban Mian,3.50,0800,1430,5
Cantonese Roast Duck,Roasted Duck Rice,3.30,0800,2030,0,1,2,3,4
Cantonese Roast Duck,Char Siew Rice,2.80,0800,2030,0,1,2,3,4
Cantonese Roast Duck,Roasted Meat Rice,2.80,0800,2030,0,1,2,3,4
Cantonese Roast Duck,Char Siew Noodle,3.30,0800,2030,0,1,2,3,4
Cantonese Roast Duck,Roasted Duck Rice,3.30,0800,1430,5
Cantonese Roast Duck,Char Siew Rice,2.80,0800,1430,5
Cantonese Roast Duck,Roasted Meat Rice,2.80,0800,1430,5
Cantonese Roast Duck,Char Siew Noodle,3.30,0800,1430,5
Western Food,American Breakfast Set,4.30,0800,1130,0,1,2,3,4
Western Food,Chicken Cutlet,4.80,0800,2030,0,1,2,3,4
Western Food,Chicken Chop,4.80,0800,2030,0,1,2,3,4
Western Food,Fish & Chips,4.80,0800,2030,0,1,2,3,4
Western Food,Beef Steak,6.30,0800,2030,0,1,2,3,4
Western Food,American Breakfast Set,4.30,0800,1130,5
Western Food,Chicken Cutlet,4.80,0800,1430,5
Western Food,Chicken Chop,4.80,0800,1430,5
Western Food,Fish & Chips,4.80,0800,1430,5
Western Food,Beef Steak,6.30,0800,1430,5
Malay BBQ,Lantong,2.80,0800,1130,0,1,2,3,4
Malay BBQ,Mee Rebus,2.80,0800,1130,0,1,2,3,4
Malay BBQ,Mee Sam,2.80,0800,1130,0,1,2,3,4
Malay BBQ,Mee Soto,2.80,0800,1130,0,1,2,3,4
Malay BBQ,Ayam Penyet,4.00,0800,2030,0,1,2,3,4
Malay BBQ,Fish Fillet,4.00,0800,2030,0,1,2,3,4
```

Figure 2: stall_menu.txt

Figure 3 shows the function we used to open the text file. It can read the text file line by line as well as split each item using commas (‘,’). It will then be converted into a directory. Inside the directory, each element will then have its own list.



```
def ReadtxtFile(self):
    self.MenuItems = {}
    StallName = ""
    self.Stallcounter = 0

    ListInStall = []

    with open("stall_menu.txt") as f_in:
        lines = f_in.readlines()
        for i in lines:
            listOfLines = i.split(',')

            ListOfDay = []
            for each in range(5, len(listOfLines)):
                ListOfDay.append(int(listOfLines[each]))

            eachItem = { "Stall": listOfLines[0],
                         "Name" : listOfLines[1],
                         "Price" : listOfLines[2],
                         "OpeningHrs" : listOfLines[3],
                         "ClosingHrs" : listOfLines[4],
                         "OperatingDays" : ListOfDay
                }

            if StallName == "" or StallName != listOfLines[0] or lines.index(i) == len(lines) - 1:
                StallName = listOfLines[0]

                if self.Stallcounter > 0:

                    if lines.index(i) == len(lines) - 1:
                        ListInStall.append(eachItem)

                    self.MenuItems["Stall" + str(self.Stallcounter)] = ListInStall
                    ListInStall = []
                self.Stallcounter += 1
            ListInStall.append(eachItem)
```

Figure 3: Def ReadtxtFile

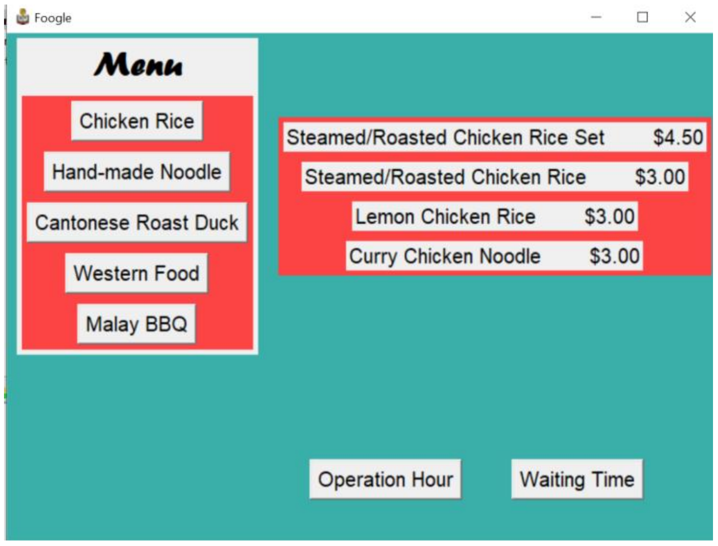
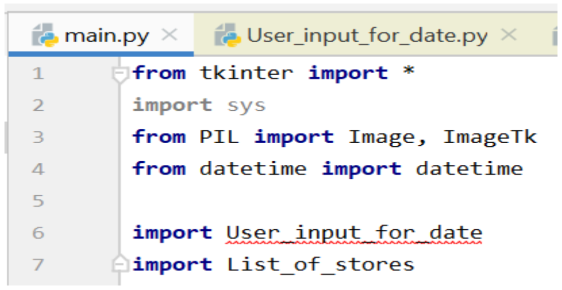


Figure 4: Menu Page

2. Display stall information based on current system date and time

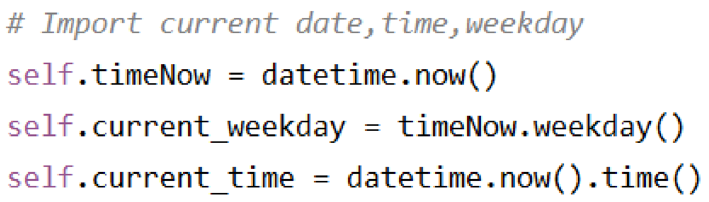
Users can view the stall information based on the current date and time. To get the current date and time, we are required to import the DateTime module as shown in figure 5 and 6.



main.py × User_input_for_date.py ×

```
from tkinter import *
import sys
from PIL import Image, ImageTk
from datetime import datetime

import User_input_for_date
import List_of_stores
```

Figure 5: Import DateTime module

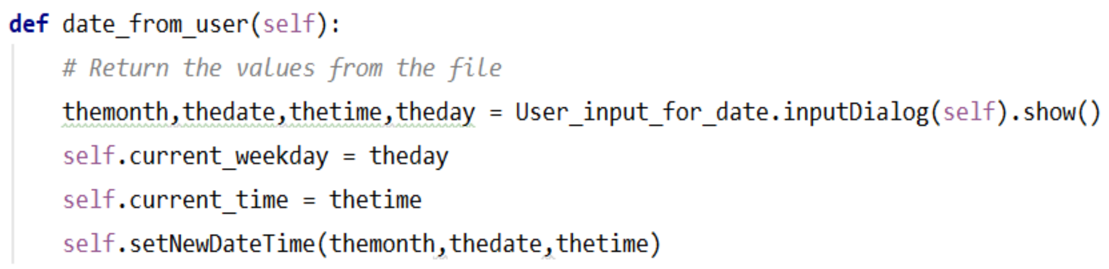


```
# Import current date,time,weekday
self.timeNow = datetime.now()
self.current_weekday = timeNow.weekday()
self.current_time = datetime.now().time()
```

Figure 6: To obtain the current date and time

3. Display stall information based on user-defined date and time

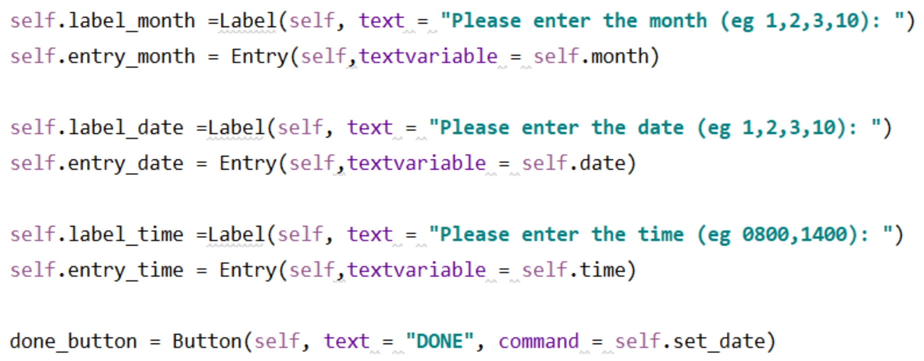
Users are also allowed to choose their desired date and time to check the stall information. We have created a new python file, especially for users to input the date and time. In figure 7, it shows how the function opens another python file, User_input_for_date.py, as well as return the values that the users have input. It will replace the 'current' date and time values.



```
def date_from_user(self):
    # Return the values from the file
    themonth,thedate,thetime,theday = User_input_for_date.inputDialog(self).show()
    self.current_weekday = theday
    self.current_time = thetime
    self.setNewDateTime(themonth,thedate,thetime)
```

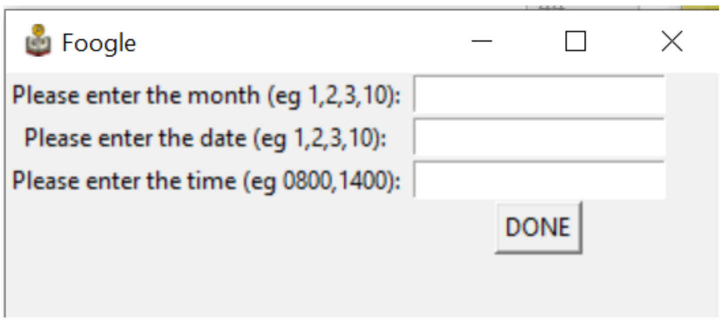
Figure 7: Function to open another User_input_for_date.py

Below coding in figure 8 is ask users to input their desired date and time using Tkinter widget.



```
self.label_month =Label(self, text = "Please enter the month (eg 1,2,3,10): ")
self.entry_month = Entry(self,textvariable = self.month)

self.label_date =Label(self, text = "Please enter the date (eg 1,2,3,10): ")
self.entry_date = Entry(self,textvariable = self.date)

self.label_time =Label(self, text = "Please enter the time (eg 0800,1400): ")
self.entry_time = Entry(self,textvariable = self.time)

done_button = Button(self, text = "DONE", command = self.set_date)
```

Figure 8: Tkinter Widget



Foogle

Please enter the month (eg 1,2,3,10):

Please enter the date (eg 1,2,3,10):

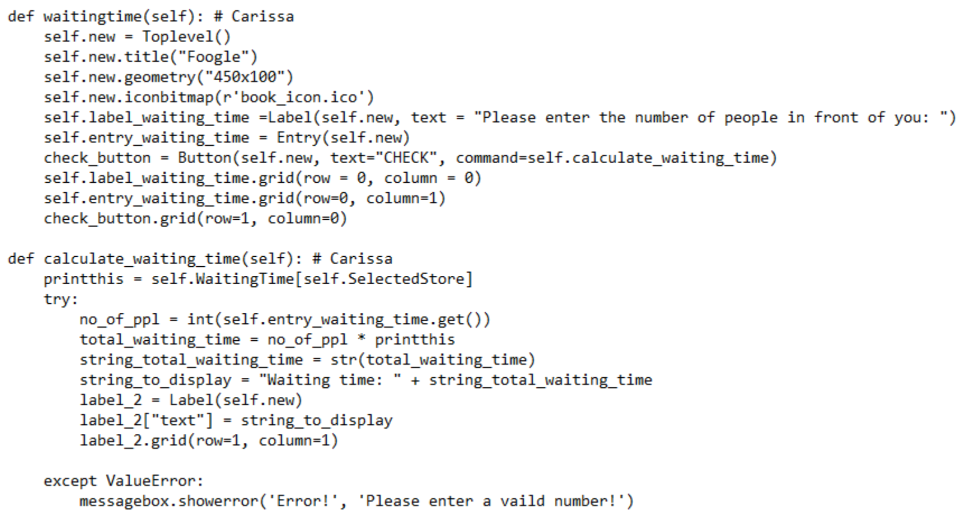
Please enter the time (eg 0800,1400):

DONE

Figure 9: Foogle: Window to enquire user's input

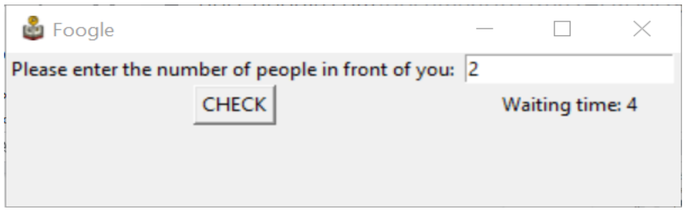
4. **Calculate estimated waiting time for the stall by asking the user to enter the number of people in the queue**

In figure 10, the both functions is used to calculate the estimated waiting based on user input. Figure 11 is the outcome of the result.



```
def waitingtime(self): # Carissa
    self.new = Toplevel()
    self.new.title("Foogle")
    self.new.geometry("450x100")
    self.new.iconbitmap(r'book_icon.ico')
    self.label_waiting_time =Label(self.new, text = "Please enter the number of people in front of you: ")
    self.entry_waiting_time = Entry(self.new)
    check_button = Button(self.new, text="CHECK", command=self.calculate_waiting_time)
    self.label_waiting_time.grid(row = 0, column = 0)
    self.entry_waiting_time.grid(row=0, column=1)
    check_button.grid(row=1, column=0)

def calculate_waiting_time(self): # Carissa
    printthis = self.WaitingTime[self.SelectedStore]
    try:
        no_of_ppl = int(self.entry_waiting_time.get())
        total_waiting_time = no_of_ppl * printthis
        string_total_waiting_time = str(total_waiting_time)
        string_to_display = "Waiting time: " + string_total_waiting_time
        label_2 = Label(self.new)
        label_2["text"] = string_to_display
        label_2.grid(row=1, column=1)

    except ValueError:
        messagebox.showerror('Error!', 'Please enter a vaild number!')
```

Figure 10: Calculate the estimated waiting time



Foogle

Please enter the number of people in front of you: 2

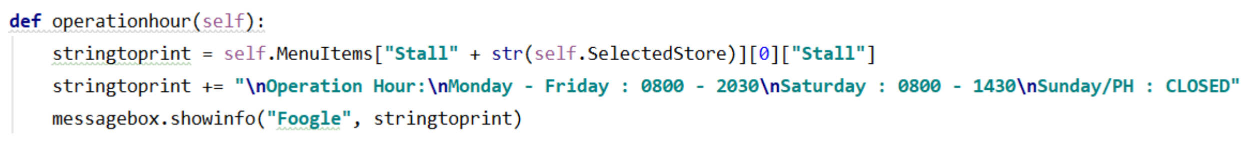
CHECK

Waiting time: 4

Figure 11: Foogle: Asking for the waiting time

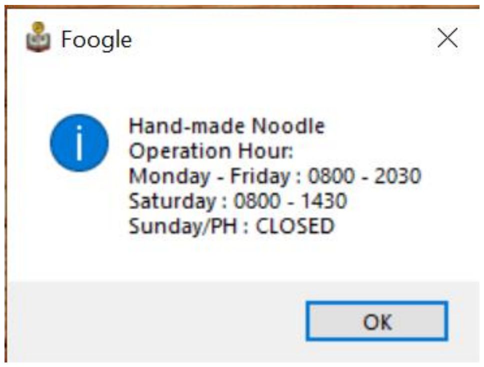
5. Allow to check the operating hours for all stalls

Users are allowed to check individual stall operating hours.



```python
def operationhour(self):
    stringtoprint = self.MenuItems["Stall" + str(self.SelectedStore)][0]["Stall"]
    stringtoprint += "\nOperation Hour:\nMonday - Friday : 0800 - 2030\nSaturday : 0800 - 1430\nSunday/PH : CLOSED"
    messagebox.showinfo("Foogle", stringtoprint)
```

Figure 12: Operation Hours

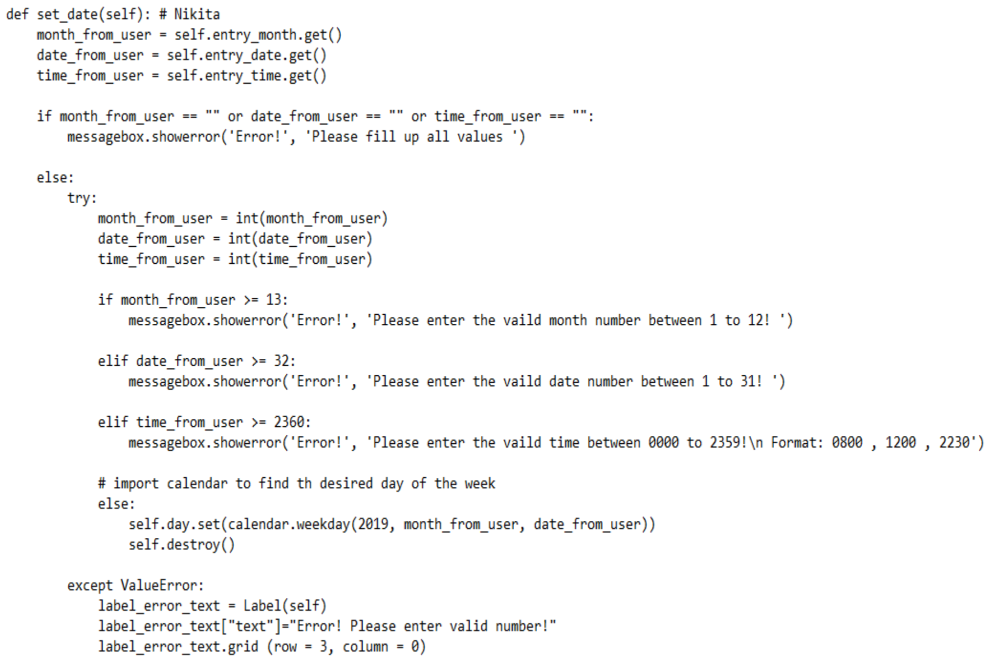


```
Hand-made Noodle
Operation Hour:
Monday - Friday : 0800 - 2030
Saturday : 0800 - 1430
Sunday/PH : CLOSED
```

Figure 13: Foogle: Operation Hour for individual stalls

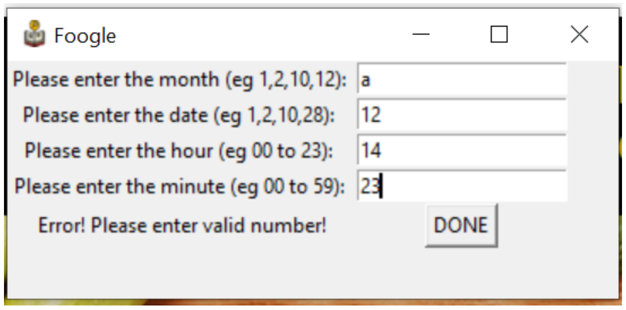
Program Testing

Exception Handling: The program will prompt error when the user input wrongly. As long as the user input alphabet or did not meet the condition, it will not allow the user to continue until it meets the requirement.



```
def set_date(self): # Nikita
    month_from_user = self.entry_month.get()
    date_from_user = self.entry_date.get()
    time_from_user = self.entry_time.get()

    if month_from_user == "" or date_from_user == "" or time_from_user == "":
        messagebox.showerror('Error!', 'Please fill up all values ')

    else:
        try:
            month_from_user = int(month_from_user)
            date_from_user = int(date_from_user)
            time_from_user = int(time_from_user)

            if month_from_user >= 13:
                messagebox.showerror('Error!', 'Please enter the vaild month number between 1 to 12! ')

            elif date_from_user >= 32:
                messagebox.showerror('Error!', 'Please enter the vaild date number between 1 to 31! ')

            elif time_from_user >= 2360:
                messagebox.showerror('Error!', 'Please enter the vaild time between 0000 to 2359!\n Format: 0800 , 1200 , 2230')

            # import calendar to find th desired day of the week
            else:
                self.day.set(calendar.weekday(2019, month_from_user, date_from_user))
                self.destroy()

        except ValueError:
            label_error_text = Label(self)
            label_error_text["text"]="Error! Please enter valid number!"
            label_error_text.grid (row = 3, column = 0)
```

Figure 14: Coding for Exception Handling



Foogle

Please enter the month (eg 1,2,10,12): a

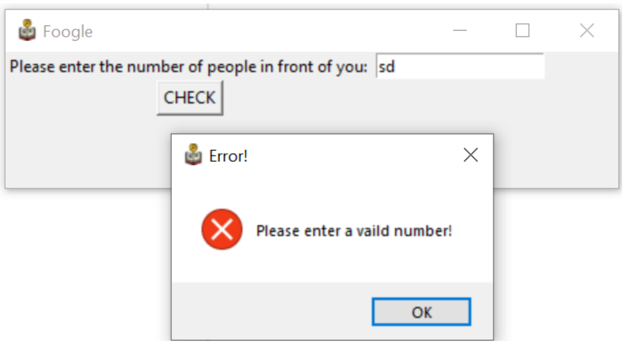
Please enter the date (eg 1,2,10,28): 12

Please enter the hour (eg 00 to 23): 14

Please enter the minute (eg 00 to 59): 23

Error! Please enter valid number!

Figure 15: Foogle: Error!



Foogle

Please enter the number of people in front of you: sd

Error!

Please enter a vaild number!

Figure 16: Foogle: Error!

Contribution of Individual Member

Member:	Contribution:
Carissa	Report: Brief Description of The Important User Defined Function, Program testing Source Code: Import and read the text file, Design of the entry widget, Display the menu information, Operation hour, Waiting time
Bachhas Nikita	Report: Abstract, Problem Statement, Solution, PowerPoint Slides Source Code: Design of the Menu page, User-defined date and time
Au Yi Xian	Report: Flow Chart, Store Information, Conclusion Source Code: stall_menu.txt, Design of the MainWindow (Root), Current date and time

Reflection: Yi Xian

During the course of this assignment, I have a better understanding of how to use the functions in python such as if-else statements and selections. This can be illustrated in our real-time canteen system whereby we consider the possible cases when a user will type in the system (i.e. the use of conditional statements, if-else) and also how the program selects the choice of food user wants (i.e. the use of list in list and dictionaries functions).
This assignment taught me that for every good algorithm design, it must comprise of steps precisely stated clearly (i.e. comments jotted down for each main step) and algorithm must come to an end after a certain number of steps.
Some of the difficulties encounters are not being able to know how to create a button, import image and the numbers to put for the grid and row for each page layout. These functions are not taught in the computational thinking module hence, a self-research must be done and watching online videos to help solve these problems.
In addition, due to the limited time and knowledge, the program created is not really user-friendly. This can be seen when running the program, after the user input the date and time it does not open a page to show the list of stalls in North Spine's canteen. Instead, users have to click on the store's menu button to see the list of stalls.
Overall, I feel that this assignment helps me better solving and understanding a problem using the abstraction method and a flowchart to gives a systematic flow on how you want the program to work. Finally, it encourages student to be a proactive learner.

Reflection: Carissa

Throughout the project, there are several challenges I have encountered. As the three of us are new to Python and Tkinter, this has made it harder for us to complete the project. Besides that, we had difficulties to coordinate with each other due to the different schedules we had. This has slowed down our progress in completing the project. One of the coding challenges I faced will be passing the variables around the functions, classes and python scripts. I tried using global variables at first but it did not work. I went to ask around other groups what did they use to pass the variables around. They suggest me to try the instance method. I did numerous research on how it works and finally, I managed to get it to run.

The GUI layout is one of the things I want to improve the most. Due to the lack of knowledge and time, we could not able to design the GUI app nicer. Next, I would also like to improve the coding for the function for operating hour.

Reflection: Nikita

Throughout the entire period of doing this assignment, I was constantly learning new information such as how to use Tkinter to create a GUI and also, revising all the topics that I have learnt during classes such as using if-elif-else statements to distinguish through user's data input. The assignment also taught me how the value of time-management as we had to keep up with the current classwork and continue to carry out the coding assignment together as a group.

One issue that I did encounter was I was unsure of how to replace the current date and time with the user's desired date and time. However, after much research, I realised that the Datetime module can be used to recognise the time inserted by the user in the 24-hour-clock system.

I also had trouble with the 24-hour clock system, as whenever the user enters an input, whose last two digits are greater than 59 we experience an error (e.g. In the 24 hour clock system you have to enter a time between 0000 and 2359. However, the issue arises if the user keys in 1178) and the program does not continue unless a new date is inserted. I managed to solve this issue by prompting to user to enter the hour and minute input separately rather than together.

One improvement that I would like to make is that the user interface and its design can be more attractive for the user.

Conclusion

In conclusion, Foogle can display the stall information based on the current date and time or users desired visit period. Furthermore, it also allowed the user to check the estimated waiting time by inputting the number of people queueing for the stall. Using Tkinter, the app is more user-friendly. Foogle is designed to be simple and clear-cut for comfortable for viewing. It displays a clear message and instructions for easy understanding.

References

1. https://www.ntu.edu.sg/AboutNTU/CorporateInfo/FactsFigures/Pages/UndergraduateStudentenrolment.aspx

2. https://www.ntu.edu.sg/AboutNTU/CorporateInfo/FactsFigures/Pages/GraduateStudentEnrolment.aspx

3. https://www.pexels.com/photo/five-white-plates-with-different-kinds-of-dishes-54455/

4. https://www.flaticon.es/icono-gratis/diccionario_917219

5. https://sites.psu.edu/emilykohler/2016/03/03/vegan-brochure-review/