**CZ1003 Project Report**

**DSAI 1 Team 7: Vaidyanathan Abhishek and Tan Wen Xiu**

## 1. Introduction

This program is designed using Graphic User Interface (GUI) Tkinter in Python. Each stall serves different food menus depending on the day. Every stall has a breakfast menu from 9am to 11am daily and a day menu for the rest of the day until the stall closes. The program is user friendly and allows the customisation of date and time so that the specialised menus that each stall serves is displayed accurately. Users can also calculate the approximate waiting time for stalls.

## 2. Algorithm Design

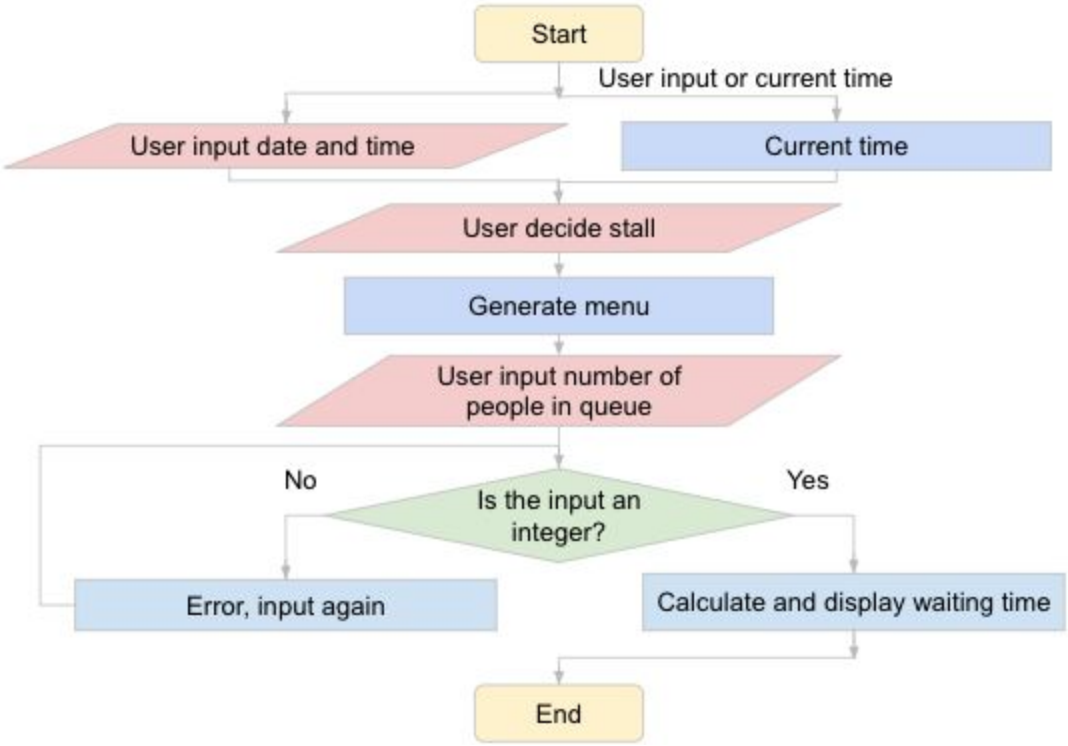


*Figure 1: A top level flowchart of the program design*

The general flowchart of the program is shown in Figure 1.


*Figure 2: The start of the program*

The program allows users to select between two buttons. The ‘user defined time’ button directs the user to the page in figure 3 to input the data and time. The ‘current time’ button determines the current time and leads the user directly to the screen in figure 5.

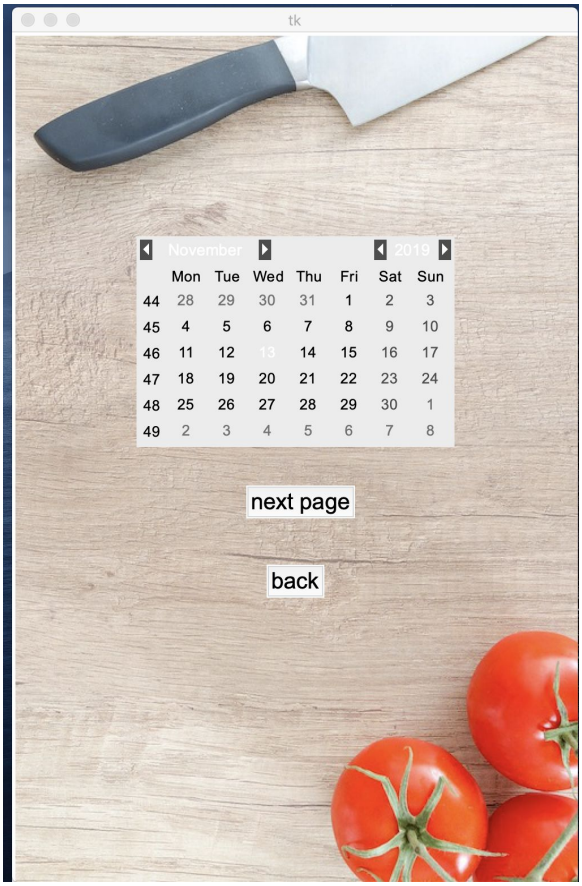


*Figure 3: User can select the date*

The user can select a date from the calendar shown in Figure 3.

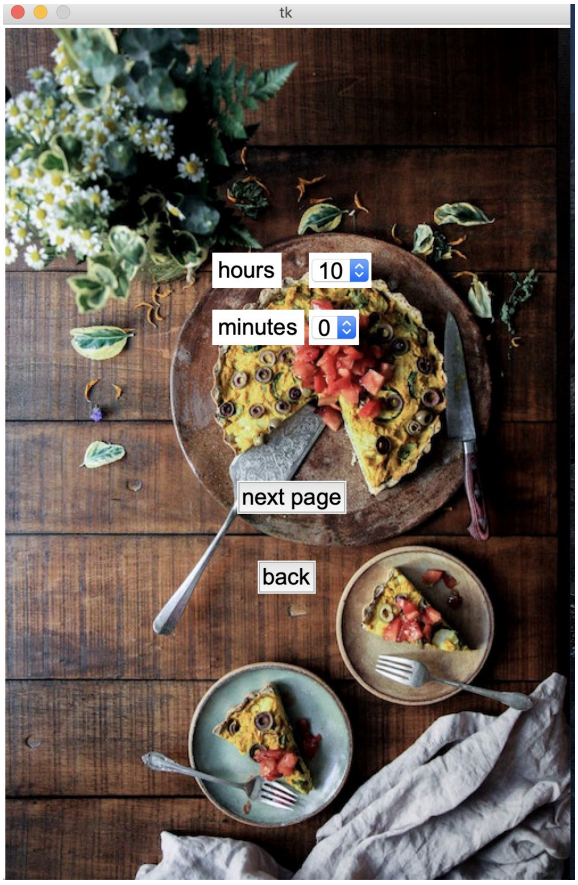


*Figure 4: Users can select the time of visit*

The dropdown for ‘hours’ includes numbers from 1 to 24 and the dropdown next to ‘minutes’ have options 15, 30 and 45 to approximate the time. The ‘next page’ button leads him to the window in Figure 5.

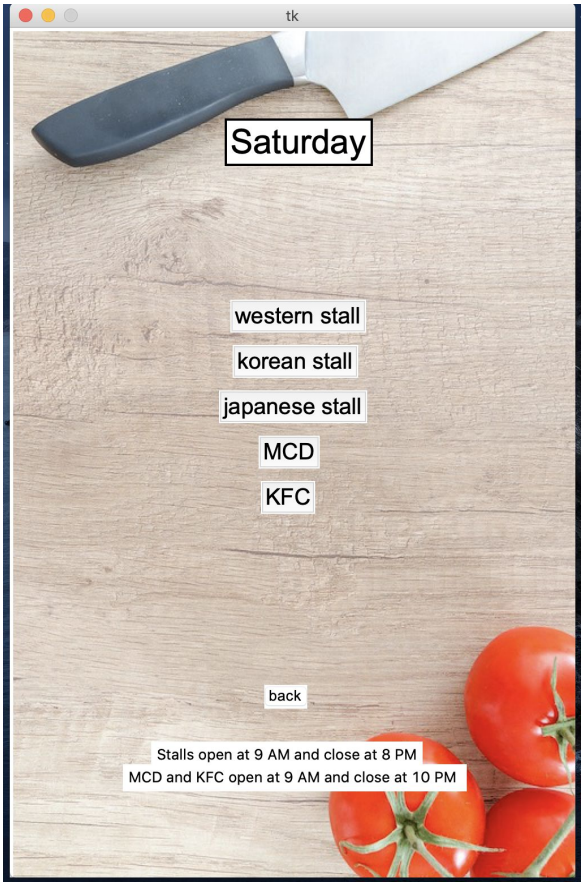


*Figure 5: The stalls in Northspine canteen are displayed in this screen*

The user can select the stall. For example, clicking the ‘western stall’ button would lead to the page shown in Figure 6.


*Figure 6: The breakfast menu served by the Western stall*

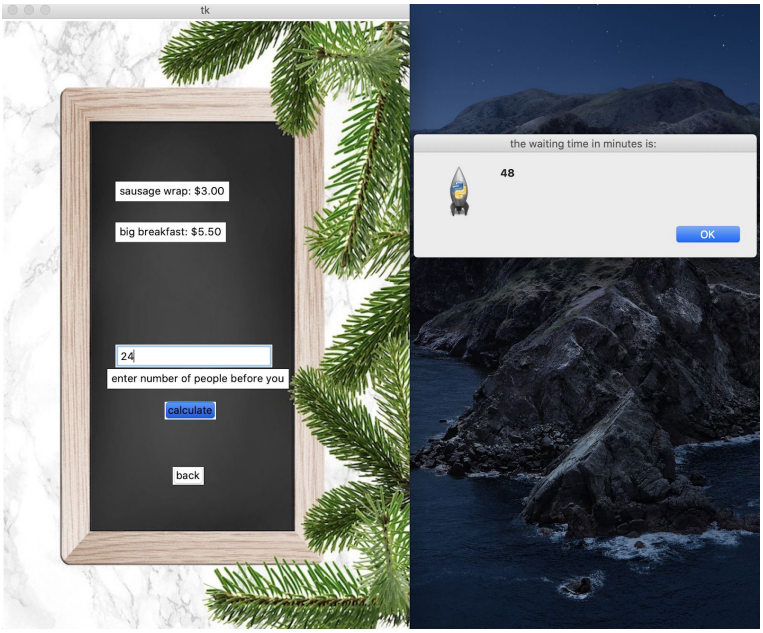


*Figure 7: Messagebox indicating waiting time in minutes*

Figure 6 displays the breakfast menu served by the Western stall on the specific day. In the textbox, the user can input a digit that indicates the number of people before him in

the queue. Clicking the ‘calculate’ button shows the approximate waiting time in a message box in Figure 7.

Every window has a ‘back’ button which allows users to return to the previous page. This allows for user friendliness as the user can amend his options. The ‘exit’ button present in Figure 2 allows users to end the program.

## 3. User Defined Functions

### 3.1. Determining the Day

```
weekdays_list=
["Monday","Tuesday","Wednesday","Thursday"
,"Friday","Saturday","Sunday"]
```

*Figure 8: Extracted code for week days*

The calendar in Figure 3 is created using the Tkcalendar module to obtain the date in the form of year, month and day from the user input. The date is changed from a string to a datetime object. The index of the day of the week is returned in the form of a list shown in Figure 8.

### 3.2. Determining the Time


*Figure 9: Breakfast menu*


*Figure 10: Day menu*


*Figure 11: Shop closed*

The dropdown menus in Figure 4 allow the user to select the time in hours and minutes. The integer of the hour is used to determine whether the breakfast or day menu should be displayed. If the input is from 8 to 11, the breakfast menu is displayed in Figure 9. If the input is 12 to 22, the day menu is displayed in Figure 10. Otherwise, the shop is

closed, seen in Figure 11. This ensures that there is no error in time input and the user gets the accurate information.

### 3.3 Determining the Number of People in the Queue (Exception Handling

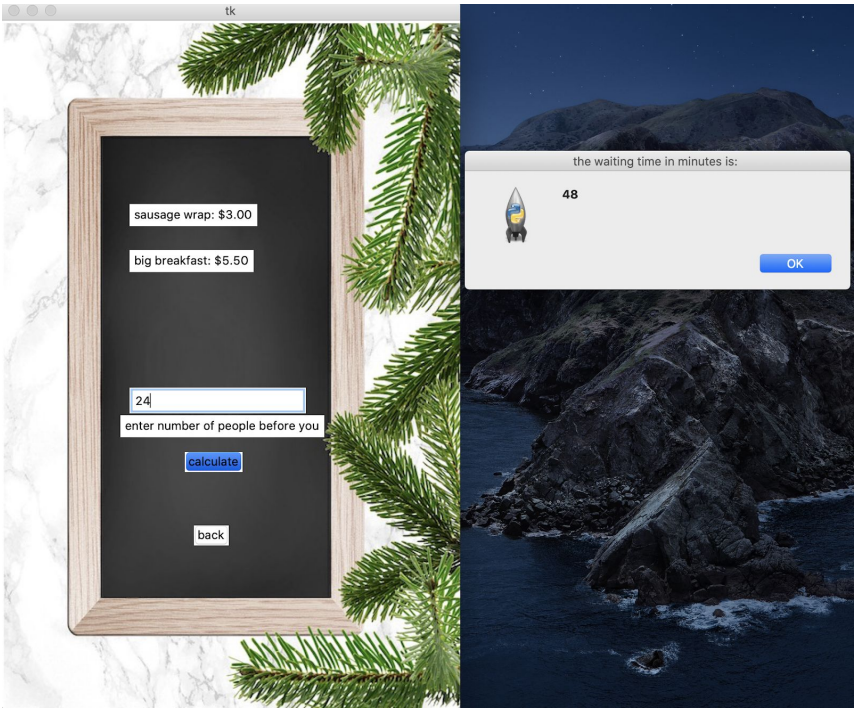


*Figure 12: Message box showing the waiting time in minutes for the user's turn in line*

A textbox allows users to enter the number of people before them in a queue. This calculates the approximate waiting time and the result will be displayed in a message box, depicted in Figure 12.

```
def ABT(s):
    try:
        s = int(s)
        if 50>= int(s) >= 0:
            s = int(s) * 2
            tkinter.messagebox.showinfo("the waiting time in minutes is: ",s)
        else:
            raise ValueError
    except ValueError:
        tkinter.messagebox.showinfo("invalid input", "please enter the correct
            number of people standing in queue")
```

*Figure 13: Extracted code for function to calculate the waiting time in queue*

To handle the errors when users input characters other than positive integer numbers below 50, a try-except statement is used, depicted in Figure 13. An error message is shown in Figures 14 to 16 when the input is invalid. The user can close the message box and try again.

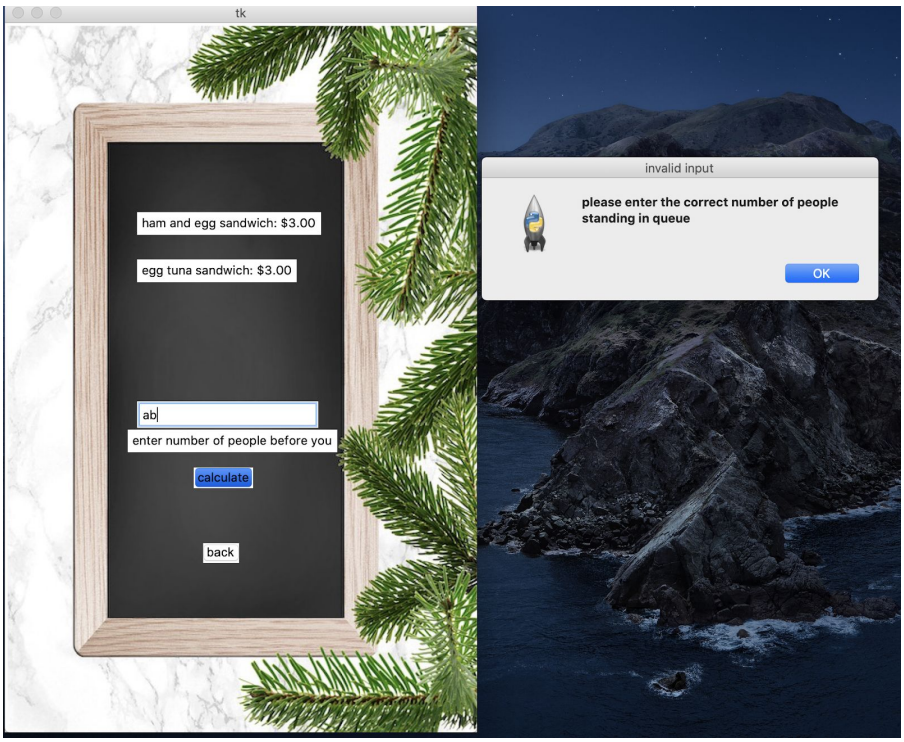


*Figure 14: Invalid input when letters are input*

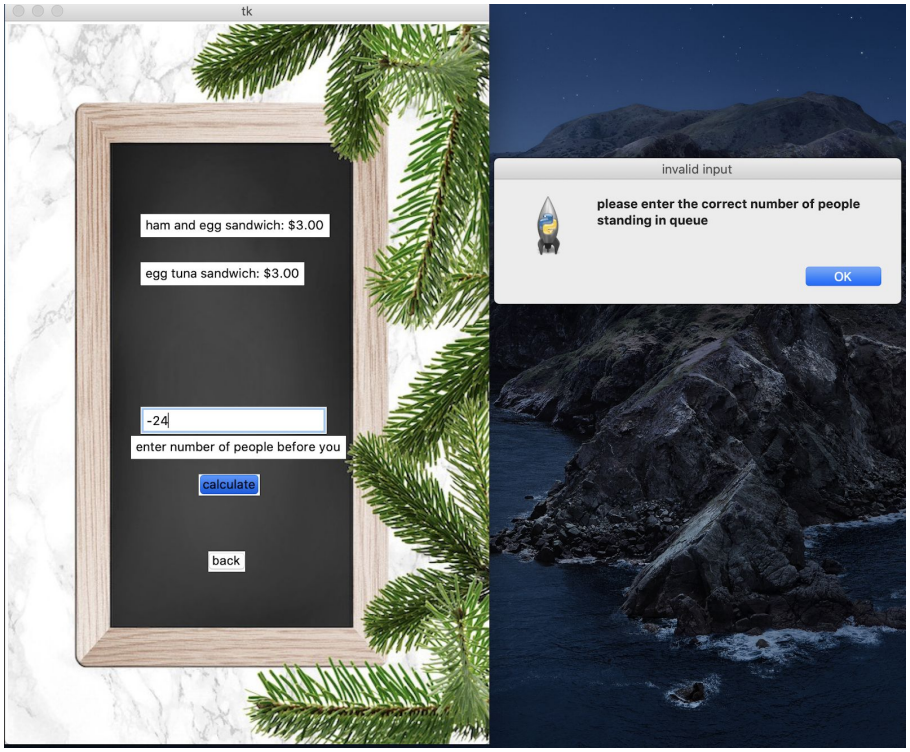


*Figure 15: Invalid input when negative integers are input*

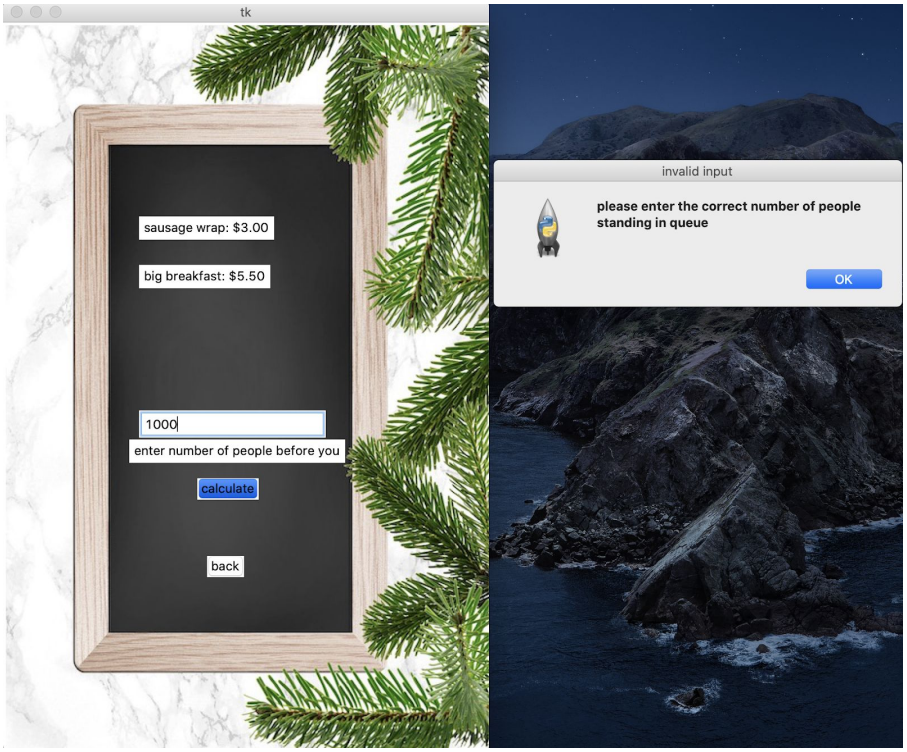


*Figure 16: Invalid input when numbers larger than 50 are input*

## 4. Data structures

```
# ======================korean food store=============================
korean_food = {'A': 'cheese ramyeon: $4.00',
               'B': 'bibimbap: $4.80',
               'C': 'hotplate chicken set: $5.00',
               'D': 'hotplate beef set: $5.50',
               'E': 'kimchi fried rice: $4.50',
               'F': 'kimchi soup: $4.00',
               'G': 'chicken ramyeon: $4.00',
               'H': 'korean pancake: $3.50',
               'I': 'tteokbokki: $3.50',
               'J': 'army stew: $4.00',
               }

#korean food offered everyday
korean_food_days = {'Monday': [korean_food['A'], korean_food['D'], korean_food['F'], korean_food['G']],
                    'Tuesday': [korean_food['B'], korean_food['E'], korean_food['G'], korean_food['C']],
                    'Wednesday': [korean_food['D'], korean_food['C'], korean_food['E'], korean_food['A']],
                    'Thursday': [korean_food['A'], korean_food['E'], korean_food['C'], korean_food['G']],
                    'Friday': [korean_food['B'], korean_food['C'], korean_food['F'], korean_food['D']],
                    'Saturday': [korean_food['A'], korean_food['C'], korean_food['D'], korean_food['F']],
                    'Sunday': [korean_food['B'], korean_food['D'], korean_food['G'], korean_food['C']]
                   }

korean_food_breakfast = {'Monday': [korean_food['I'], korean_food['H']],
                         'Tuesday': [korean_food['H'], korean_food['J']],
                         'Wednesday': [korean_food['I'], korean_food['H']],
                         'Thursday': [korean_food['H'], korean_food['J']],
                         'Friday': [korean_food['I'], korean_food['H']],
                         'Saturday': [korean_food['I'], korean_food['J']],
                         'Sunday': [korean_food['H'], korean_food['J']]
                        }
```

*Figure 17: Extracted code showing the use of dictionaries and lists in the food menus*

Dictionaries and lists were used to create the food menus for each stall, an example is shown in Figure 18. All the food menus are contained in the dictionary “korean_food”. Two separate dictionaries, “korean_food_days” and “korean_food_breakfast” are created to differentiate the breakfast from the day menu. The keys are the days and the values are a list of food retrieved from the dictionary “korean_food”. For example, if the key ‘Monday’ is called, then the values of ‘Monday’ are returned.

## 5. Member Contribution

| ABHISHEK | WEN XIU |
|---|---|
| Creating GUI using Tkinter in Python | Creating food menus using dictionary |
| Creating calendar in Tkinter | Creating function to retrieve menus |
| Creating user input functions | Writing report |
| Creating function for queue time | Creating presentation slides |
| Layout and design of GUI | |
| Writing report [ explanation of functions and reflections] | |

## 6. Reflection

As a team with only 2 people who did not have prior coding experience, we realized that we were disadvantaged and to complete more work individually compared to other teams. We faced many challenges while working on this project.

Firstly, we wanted to create a user input time. Initially, we wanted to allow the user to type the time in the format “HHMM” in a text box to retrieve the relevant menus for the stall at that time. However, there were limitations to this format, due to the ‘hour’ format being only 1 to 24, and the ‘minute’ format being only 0 to 59, hence, this method was very complicated for us. We tried to use the datetime module to format the time, however, we did not find a solution to easily create a Boolean statement to fit into our function to retrieve the relevant menu for the time periods. In the end, we decided to get

the user input time through drop down menus to simplify the process and minimise user input error.

Another challenge was to get the user input for date. The exception handling for this input is too tedious for us as it requires to check if the user's inputs for year, month and day are correct. We then decided to use drop down menus instead. However, this involved too many if-else statements depending on the year, which affected the number of days in the month. For example, the number of days in February changes according to the leap year system. There are also different number of days for every month. Finally, we used the calendar function in tkinter which was challenging, but nevertheless, it was simpler than the other two methods. The only problem we faced was to retrieve the date that the user selects and convert that to a datetime object to get the day number of the week. However, we solved it by using the dateutil module we imported.

This course equipped both of us with the basic knowledge required to code a program so that we could complete this project. We learnt basic python functions and implemented them in the process of creating this program. More importantly, the project helped us understand and troubleshoot errors that we encountered while coding. For example, this project helped us understand the type of errors which we can use try-except statements, and the conditions that we could make use of if-elif-else statements. Furthermore, we have learnt to utilise lists and dictionaries to store data. On top of that, we have walked away with new knowledge and skills in coding with Tkinter in Python since we used it as our GUI for this project. Learning Tkinter was a challenge since many of the functions were unknown to us previously, hence we had to do plenty of research before we could understand the basic functions and how to use them in our code. Overall, the project helped us gain knowledge about the types of functions to use, different errors that might occur while coding and helped us understand how we can simplify the program to get the relevant output that we require.

Further improvements could be made to this program to allow greater flexibility in user input and user choices, such as to include more food stalls in the program. Different types of functions to estimate the different waiting time for each stall could also be made to take into account the difference in speed of serving customers for each stall.

## 7. References

Bernd Klein, Bodenseo, “Python Tkinter”, Python Course, Last modified 2019. [Internet] https://www.python-course.eu/python_tkinter.php

Mark Roseman, “Tkdocs”, Tkdocs, Last modified 2019. [Internet] https://tkdocs.com/tutorial