SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

PROJECT TITLE : Real-time NTU North Spine Canteen Information System

Group Members :

RAYMOND GOH KANG SHENG

(U1921489J)

S SRI KALKI

(U1921575L)

OWEN CHIANG YU LIANG

(U1920496J)

LAB GROUP : FS5

TUTOR : ASSOC PROF VUN CHAN HUA, NICHOLAS

CZ1003-INTRO TO COMPUTATIONAL THINKING

Mini - Project

AY 2019-20 SEMESTER 1

DATE OF SUBMISSION :

12 November 2019

Objective

A menu system for NTU north spine canteen with graphical user Interface with several functionalities.

Requirement

A. Store and display stall information
B. Store and display stall menus
C. Display stall information and menus based on current system date and time
D. Display stall information and menus based on user defined date and time
E. Calculate estimated waiting time for the stall by asking user to enter the number of people in the queue
F. Allow to check the operating hours for all stalls

Tasking and Responsibilities

Raymond:

- Development of the Tkinter interface
- Development of part E and F
- Development of Text to Speech feature
- Deployment of Web Scraping Technique using beautifulsoup4
- Optimizing the application with Threading

Kalki:

- Development of part A and B
- Development of the file handling system for the application using Pickle

Owen:

- Development of part C and D
- Enhancement of Tkinter Interface

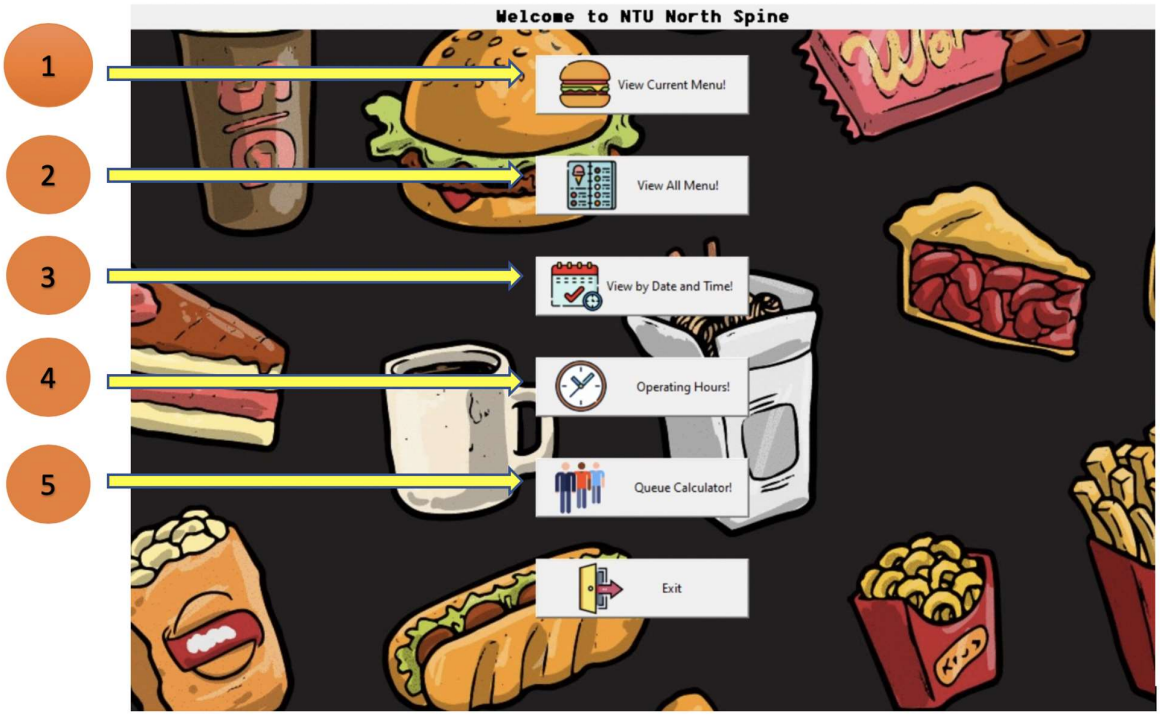
Specification

- The Graphical User Interface is based and written using Tkinter toolkit.
- External libraries were used to support some of the functionality of the application:
    - Tkcalendar
    - Pygame
    - Pyttsx3
    - BeautifulSoup4

- Internal libraries:
    - Datetime
    - RE
    - Tkinter
    - Calendar
    - Pickle
    - Threading

Top Module Flowchart

The overall flow of the program



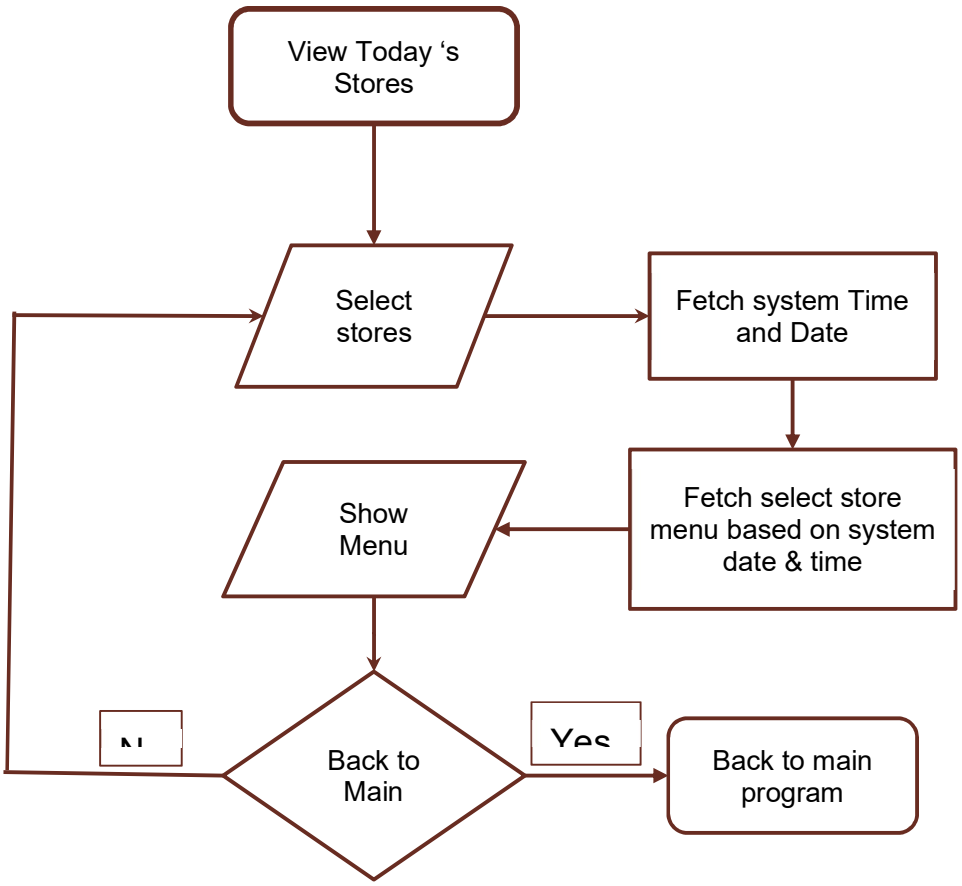
Main menu of the application

This application is designed such that users find it easy to use and pleasurable to use. The flow of the application is intuitive, controls are meaningful, and outputs are clear to see.

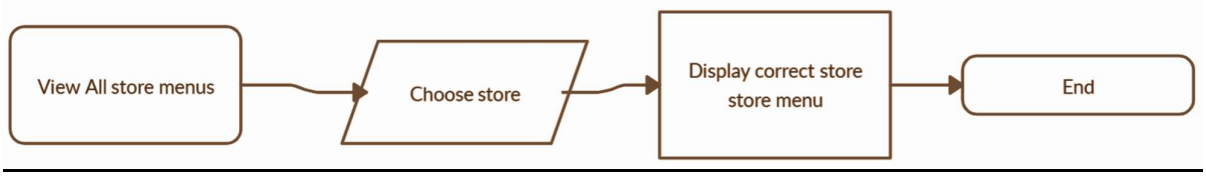
It is also accompanied by a voice assistant, where it provides the users with meaningful instructions to follow. 5 features are included in the application according to the requirement of the project.

Flowcharts

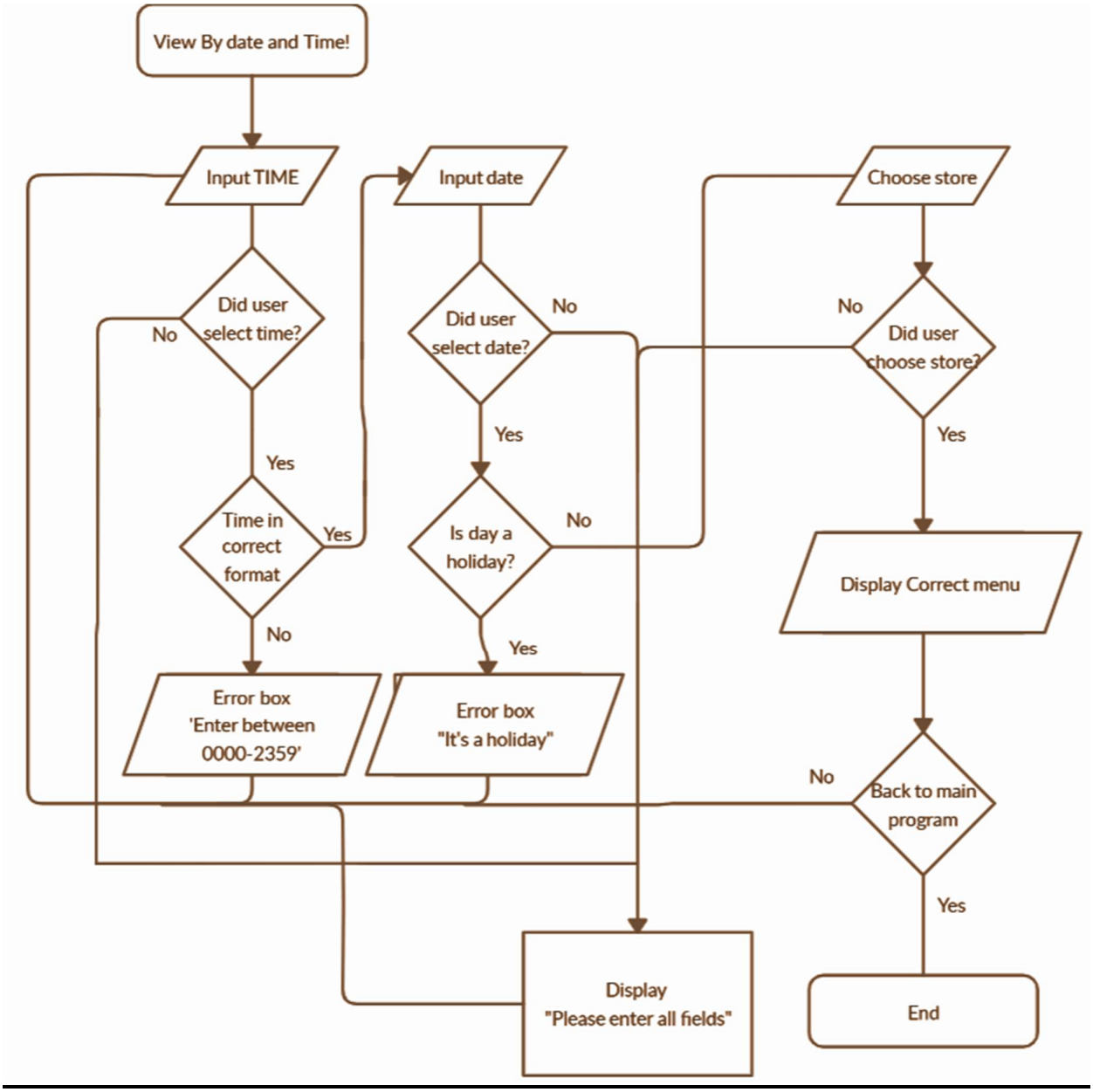
1. View current menu



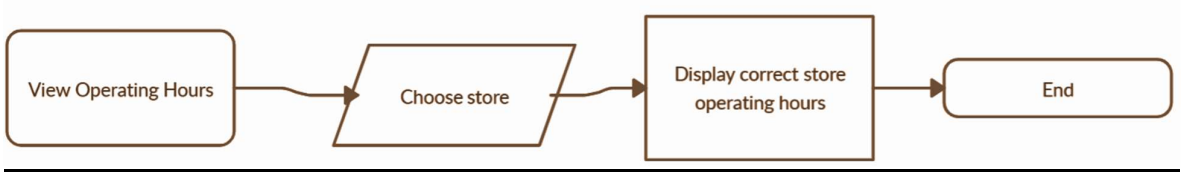
2. View all menu



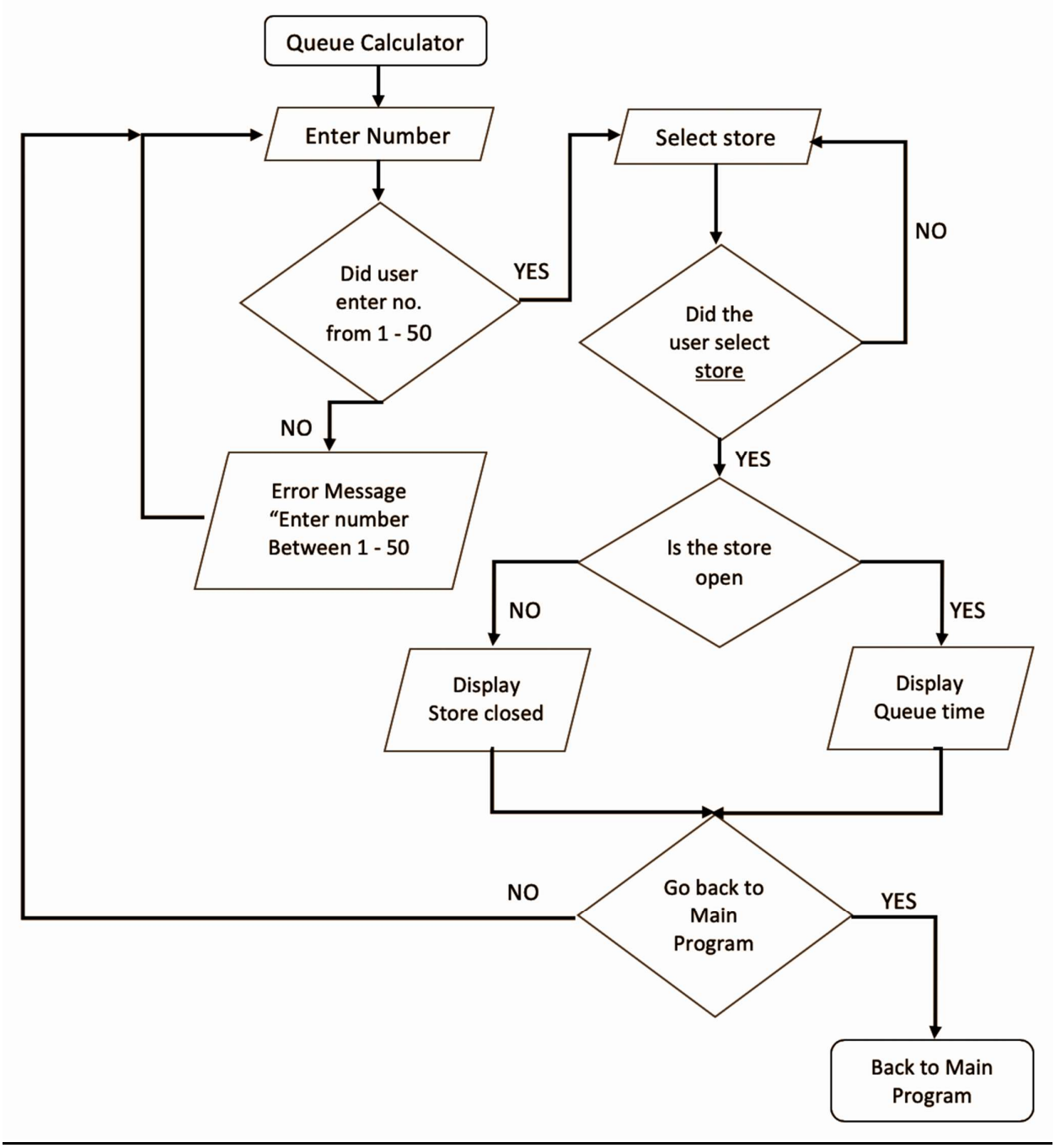
3. View by date and time



4. Operating Hours



5. Queue calculator



User-Defined Functions

1. File Handling

```
# CREATED BY HARISH
# FUNCTION TO FETCH MENU FROM TXT
def read_menu():
    file_name = "menu.txt"
    fileObject = open(file_name, 'rb')
    global menu
    menu = pickle.load(fileObject)
    fileObject.close()
    return menu

read_menu();
```

open and read .txt

load data and close .txt

```
# CREATED BY HARISH
# FUNCTION TO FETCH STORE INFORMATION FROM TXT
def read_store_data():
    file_name = "store.txt"
    fileObject = open(file_name, 'rb')
    global store_list
    global operating_hours1
    global queue_time_calculator
    global operating_hours2
    global menu_switch_time
```

open and read .txt

```
    store_list = pickle.load(fileObject)
    operating_hours1 = pickle.load(fileObject)
    queue_time_calculator = pickle.load(fileObject)
    operating_hours2 = pickle.load(fileObject)
    menu_switch_time = pickle.load(fileObject)
    fileObject.close()
    return menu_switch_time, store_list, operating_hours1, queue_time_calculator, operating_hours2
```

load data and close .txt

Our team uses pickle for file handling; the above code will open, read, and load data from **menu.txt**, which consists of store information and menu.

2. View Current Menu

```
def assign_store(selection):
    print(selection)

    current_time = int((dt.datetime.now()).strftime("%H%M"))
    current_day= (dt.datetime.now()).strftime("%A")
    if (current_day == 'Tuesday'or current_day == 'Thursday'or current_day == 'Saturday'):
        oddeven = 'even'
    else:
        oddeven= 'odd'

    if(int(menu_switch_time[selection])> current_time):
        am_pm_select= 'am'
    else:
        am_pm_select= 'pm'

    if int(operating_hours2[selection][current_day]['Opening'])<= current_time < int(operating_hours2[selection][current_day]['Closing']):
        print(menu[selection][oddeven][am_pm_select])
        show_menu = menu[selection][oddeven][am_pm_select]
    else:
        show_menu = 'Store is currently closed \n store operating hours are\n' + operating_hours1[selection]

    op_display = tk.Label(text = show_menu , font=('Fixedsys', 20, "bold"))
    op_display.place(relheight = 0.4, relwidth = 0.6, relx = 0.2, rely = 0.20)
```

Get current system time

Get current system Date

Tues Thurs Sat is even menu

Mon Wed Fri Sun is odd menu

Check AM or PM menu

If time is in opening hours, print correct menu

If store closed, print store opening hours

Display the correct output in the GUI

Our program fetches the system time with the **datetime.now** function. It will then fetch the correct menu and display it in the GUI. If the store is closed, the operating hours of that store will be printed instead. There is only one drop-down menu so the user cannot enter the wrong input.

3. View all menu

```
def assign_store(selection):
    print(selection)

    show_menu1 = 'Odd day breakfast\n' +menu[selection]['odd']['am']
    show_menu2 = 'Odd day afternoon\n' +menu[selection]['odd']['pm']
    show_menu3 = 'Even day breakfast\n' +menu[selection]['even']['am']
    show_menu4 = 'Even day afternoon\n' +menu[selection]['even']['pm']

    op_display = tk.Label(text = show_menu1 , font=('Fixedsys', 12, "bold"))
    op_display.place(relheight = 0.3, relwidth = 0.4, relx = 0.1, rely = 0.20)

    op_display = tk.Label(text = show_menu2 , font=('Fixedsys', 12, "bold"))
    op_display.place(relheight = 0.3, relwidth = 0.4, relx = 0.5, rely = 0.20)

    op_display = tk.Label(text = show_menu3 , font=('Fixedsys', 12, "bold"))
    op_display.place(relheight = 0.3, relwidth = 0.4, relx = 0.1, rely = 0.50)

    op_display = tk.Label(text = show_menu4 , font=('Fixedsys', 12, "bold"))
    op_display.place(relheight = 0.3, relwidth = 0.4, relx = 0.5, rely = 0.50)
```

The user will choose the store with the drop-down menu. The program will then fetch the correct store with the **menu[selection]** function and display in the GUI. Each store has 4 different menus. The drop down menu will prevent the user from entering wrong inputs.

4. View by date and time

This function is to catch invalid input of the user and fetch the menu based on the selected store, time and date.

5. View store operating hours

```
# DONE BY RAYMOND GOH
# DISPLAYING OPERATING HOURS BASED ON THE STORE SELECTED VIA DROPDOWN MENU
def assign_store(selection3):
    op_hours =operating_hours1[selection3]
    op_display = tk.Label(text = op_hours , font=('Fixedsys', 20, "bold"))
    op_display.place(relheight = 0.4, relwidth = 0.6, relx = 0.2, rely = 0.20)
```

The user selects the drop-down menu and chooses the store and the selected store operating hours will be displayed. Drop-down menu will prevent the user from entering a wrong input, and data will be fetched from our **operating_hours1** dictionary.

6. Queue calculator

```
# CREATED BY RAYMOND GOH
# FUNCTION TO FETCH THE RESPECTIVE WAITING TIME IN MINS BASED ON THE STORE SELECTED
def countit(selection):
    global queue_time
    queue_time = int(queue_time_calculator[selection])
    global storesel
    storesel= selection
    return queue_time
```

This function simply matches the selection from the user to the key of the **queue_time_calculator** dictionary and fetching the value of the items to **queue_time**, and assigning selection to **storesel**.

```
# CREATED BY RAYMOND GOH
# FUNCTION TO LIMIT THE ENTRYBOX TO ONLY NUMERICAL CHARACTERS
class only_int(tk.Entry):
    def __init__(self, master=None, **kwargs):
        self.var = tk.StringVar()
        tk.Entry.__init__(self, master, textvariable=self.var, **kwargs)
        self.old_value = ''
        self.var.trace('w', self.check)
        self.get, self.set = self.var.get, self.var.set
    def check(self, *args):
        if self.get().isdigit():
            # THE CURRENT VALUE IS ONLY DIGITS; ALLOW THIS
            self.old_value = self.get()
        else:
            # THERE'S NON-DIGIT CHARACTERS IN THE INPUT; REJECT THIS
            self.set(self.old_value)
```

This function will ensure that **only numerical values** are allowed in the text box.

```
def result():
    try:
        # CATCHING USER IF THEY ENTER VALUE LESS THAN 1 AND MORE THAN 50
        if int(queue_input.get()) > 50 or int(queue_input.get()) < 1:
            messagebox.showerror("Input Error", "Please input value between 1 to 50!")
            # RESET THE ENTRYBOX VALUE TO 1
            queue_input.set("1")
            return None
        queue_input_out = queue_time * int(queue_input.get())
    # CATCHING USER IF STORE NOT SELECTED
    except NameError:
        messagebox.showerror("Input Error", "Please select store!")
        return None
    # CATCHING USER IF NO VALUE IS BEING INPUT
    except:
        messagebox.showerror("Input Error", "Please input value!")
        return None
```

This is to catch the users for invalid inputs if inputs are valid, the queue time per customer (in mins) will be multiplied with the numerical input value, the result will be assigned to **queue_input_out**. (refers to **Test Cases For Part F** for more information)

```
### COMPUTATION OF THE TOTAL QUEUE TIME ACCORDING TO THE STORE SELECTED WITH LENGTH OF THE QUEUE ###
# FETCHING CURRENT SYSTEM TIME
now = dt.datetime.now()
# ADDING NUMBER OF MINS TO CURRENT SYSTEM TIME ACCORDING TO LENGTH OF QUEUE
now_plus = now + dt.timedelta(minutes = queue_input_out)
# FORMATTING THE TIME OUTPUT
now_final = int(now_plus.strftime('%H'))*100+int(now_plus.strftime('%M'))
now_final_covt = str(int(now_plus.strftime('%H'))*100+int(now_plus.strftime('%M')))
# ADDING A COLON IN THE TIME FORMAT
if len(now_final_covt) == 4:
    now_final_covt = now_final_covt[:2] + ':' + now_final_covt[2:]
else:
    now_final_covt = now_final_covt[:1] + ':' + now_final_covt[1:]
# STRING TO DISPLAY OUTPUT BACK TO THE USER
final_form = "Estimated queuing time will be\n" + str(queue_input_out) + " mins\n" \
             + "Join the queue now to place order at\n" + now_final_covt
```

This set of codes is to fetch the current system time, adding the minutes from **queue_input_out** and formatting it accordingly for the output.

```
def display_good():
    display = tk.Label(header_frame, text = final_form , font=('Fixedsys', 16, "bold"))
    display.place(relheight = 0.2, relwidth = 0.6, relx = 0.2, rely = 0.40)
#CODE FOR THE TEXT TO SPEECH FUNCTION TO READ THE TOTAL QUEUE TIME IN MINS
def voice_good():
    engine = pyttsx3.init()
    engine.setProperty('rate',180)  #180 words per minute
    engine.setProperty('volume',0.9)
    engine.say("Estimated queuing time will be\n" + str(queue_input_out) + " mins\n")
    engine.runAndWait()
```

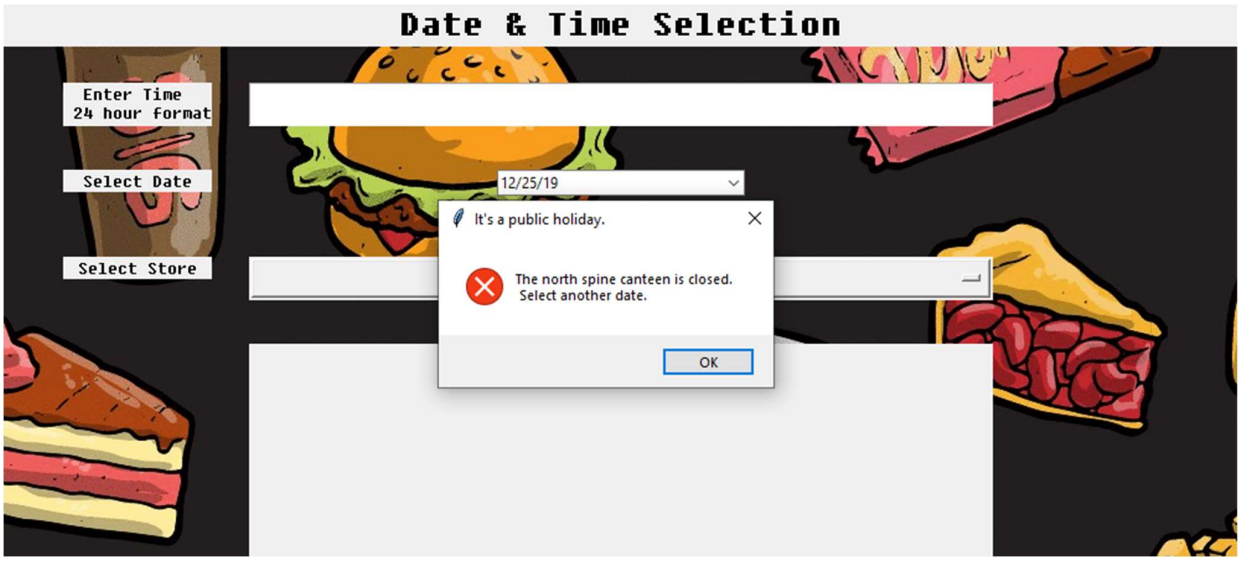
This set of codes will allow the output of the previous code to be displayed and read to the user.

Test Cases for Part D (Menu Based on Time and Date)			
Test Case	Scenario	Result	Pass/Fail
1	When the user did not enter all or any of the 3 the required fields	An error message will appear, prompting the user to enter all required information (Appendix 1.1)	Pass
2	When the user did not enter time between 0000 to 2359	An error message will appear, promoting the user to correct the time input (Appendix 1.2)	Pass
3	When the user selected a date, time and store, but the selected store is close during that defined period	The output will tell the user that the store is currently closed, and show the user the operating hours of the selected store (Appendix 1.3)	Pass
4	When the user selected a date that fails in the public holiday	An error message will appear, prompting the user to select another date, and the date input (tkcalendar) will reset to the system's date (Appendix 1.4)	Pass
5	When the user input a correct time within the operating hours of the selected store, selected a date that is not a public holiday and selected a store	The output will show the user the menu based on user's defined store, time and date (Appendix 1.5)	Pass

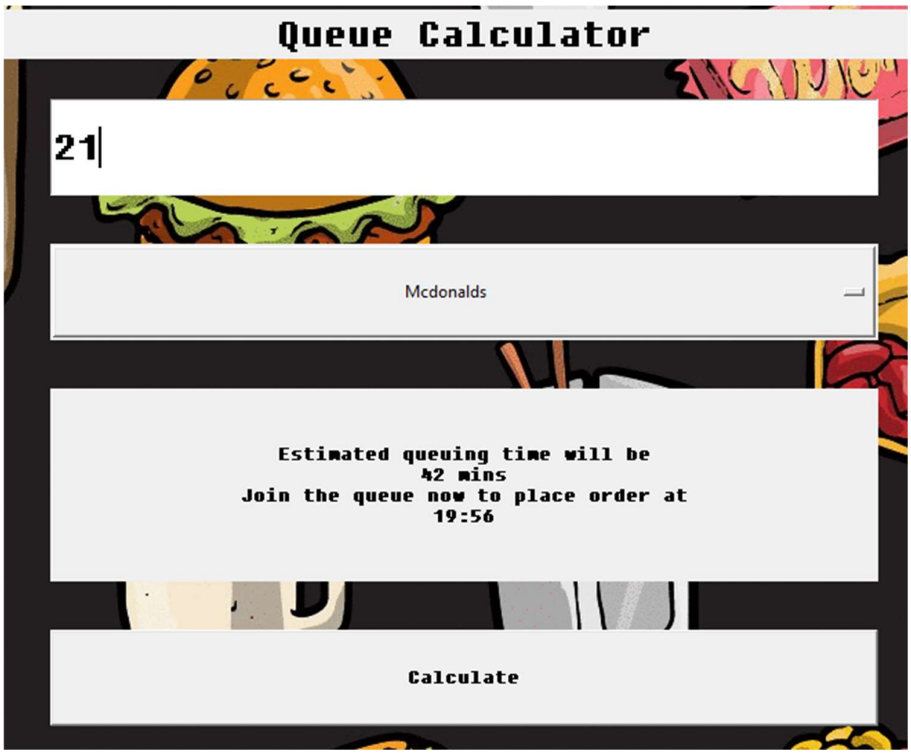
Test Cases for Part E (Queue Calculator)			
Test Case	Scenario	Result	Pass/Fail
1	User did not input any values and select any store	An error message will appear, prompting the user to input a value (Appendix 2.1)	Pass
2	User attempts to input non-numerical into the text box	The “only_int” function will ensure that only numerical character is allowed in the text box, and simply rejects any symbol and non-numerical character	Pass
3	When user input the right numerical value, but did not select any store from the optionmenu	An error message will appear, prompting the user to select a store (Appendix 2.2)	Pass
4	When store is selected but the user input numerical value that is less then 1 or more than 50	An error message will appear, prompting the user to correct the input, at the same time, the value of the text box will self-correct to “1”, preventing the same error from occurring again. (Appendix 2.3)	Pass
4	After the user key in the right numerical value and selects a store, but the selected store is current is closed or when the queue time exceeds the operating hours of the store	The output will simply warn the user that the store is currently closed, or it is too late to join the queue, as the store will be closed by then. (Appendix 2.4)	Pass
5	When the user enters the correct numerical value and selected a store	The output will show the user the duration (in mins) of the queue and the time when the user is able to make the order (Appendix 2.5)	Pass

Extra Features

Some of the extra features include:

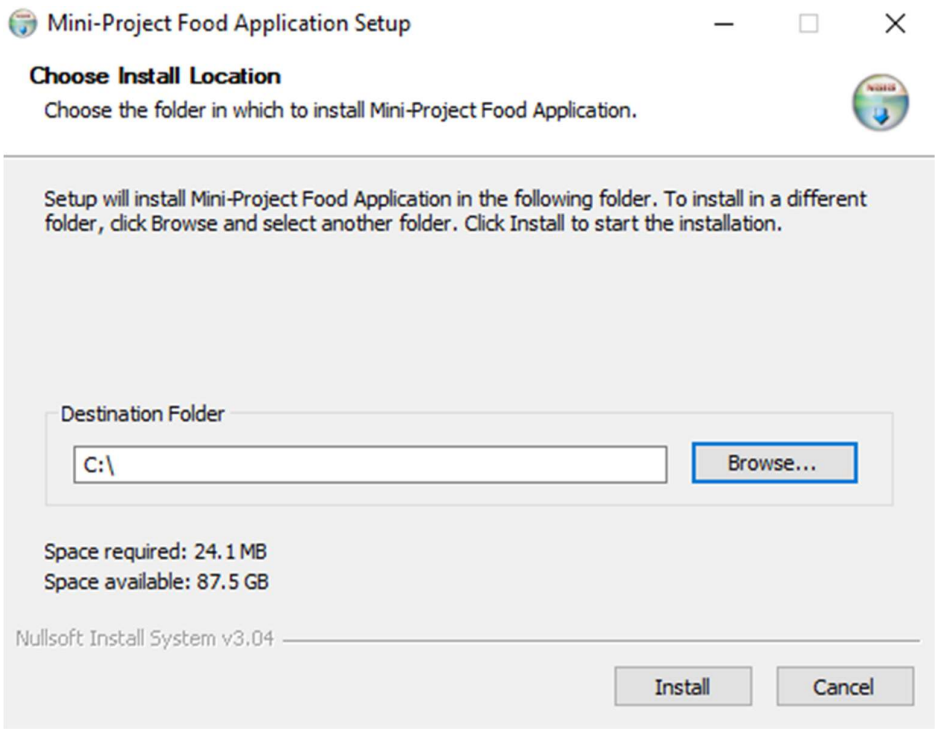


1. Able to tell the user the canteen is closed during the public holiday (Data fetch from Ministry of Manpower using beautifulsoup4)



2. Able to provide the user with not only the waiting time in minutes (Queue Calculator Feature) but also able to give them a time and telling the user is too late to join the queue if the store is closed when it's their turn to order.

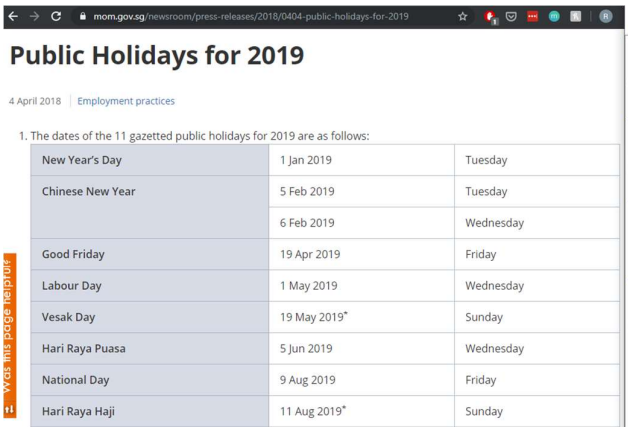
3. Voice Assistant (text to speech using ppttsx3 and pygame) to aid the user with navigation and reading the total waiting time for the Queue Calculator feature.



4. Compile the application and all its dependencies into a single package. The user can run the packaged app without installing a Python interpreter or any modules.

Function for harvesting list of public holiday from MOM site:

```
# CREATED BY RAYMOND GOH
# FUNCTION TO HARVEST DATA FROM MOM'S WEBSITE TO FETCH PUBLIC HOLIDAY LIST
def fetch_from_mom():

    url = "https://www.mom.gov.sg/newsroom/press-releases/2018/0404-public-holidays-for-2019"
    # CONNECT THE PROGRAM WITH MOM'S WEBSITE
    page = urllib.request.urlopen(url)

    soup = BeautifulSoup(page, 'html.parser')

    ph_list = []
    # FOR LOOP TO FETCH ALL DATA FROM THE TABLE OF THE SITE
    for link in soup.findAll('td'):
        if "*" in link.text:
            link = link.text.replace("*", "") # REMOVE ASTERISKS FROM SOME OF THE DATES
            ph_list.append(link)
        else:
            ph_list.append(link.text)

    global holiday_list

    # FOR LOOP TO GET ONLY DATE OF PUBLIC HOLIDAY FROM THE TABLE
    # APPEND ALL PUBLIC HOLIDAY TO A LIST
    for element in ph_list:
            if not element.isalpha():
                if "2019" in element:
                    holiday_list.append(datetime.strptime(element, '%d %b %Y').date())
```



mom.gov.sg/newsroom/press-releases/2018/0404-public-holidays-for-2019

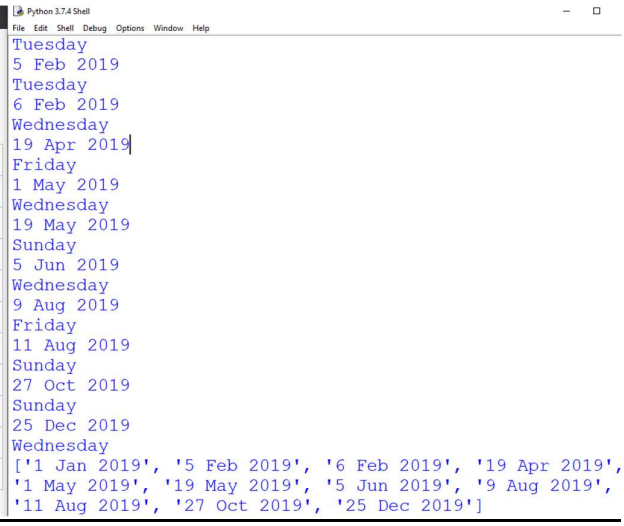
Public Holidays for 2019

4 April 2018 | Employment practices

1. The dates of the 11 gazetted public holidays for 2019 are as follows:

New Year's Day	1 Jan 2019	Tuesday
Chinese New Year	5 Feb 2019	Tuesday
	6 Feb 2019	Wednesday
Good Friday	19 Apr 2019	Friday
Labour Day	1 May 2019	Wednesday
Vesak Day	19 May 2019*	Sunday
Hari Raya Puasa	5 Jun 2019	Wednesday
National Day	9 Aug 2019	Friday
Hari Raya Haji	11 Aug 2019*	Sunday
Deepavali	27 Oct 2019*	Sunday
Christmas Day	25 Dec 2019	Wednesday

Was this page helpful?



```
Python 3.7.4 Shell
File  Edit  Shell  Debug  Options  Window  Help
Tuesday
5 Feb 2019
Tuesday
6 Feb 2019
Wednesday
19 Apr 2019
Friday
1 May 2019
Wednesday
19 May 2019
Sunday
5 Jun 2019
Wednesday
9 Aug 2019
Friday
11 Aug 2019
Sunday
27 Oct 2019
Sunday
25 Dec 2019
Wednesday
['1 Jan 2019', '5 Feb 2019', '6 Feb 2019', '19 Apr 2019',
'1 May 2019', '19 May 2019', '5 Jun 2019', '9 Aug 2019',
'11 Aug 2019', '27 Oct 2019', '25 Dec 2019']
```

Personal Reflection

Raymond

The first challenge I faced was developing the GUI of the project, as it was beyond the curriculum. However, I know that we must always be a self-directed learner. Therefore, I spent my recess week developing the GUI so that my teammates can focus on their building their features while I integrate their work into the GUI.

Teamwork is essential to the speed and quality of development, other than looking into online resources; I realized that together, we could accomplish more through project discussions. The majority of our problems were solved together.

One key concept that I inherited from the course is divide and conquer, whereby a bigger problem can be solved by first breaking it into smaller parts. This enables me to look at problems from a different perspective. Once we tackled the smaller problems, it would now be more comfortable to conquer the bigger problem. By applying this fundamental concept, along with the useful complementary lectures provided by the course, I was able to deploy complex functions used in the project.

Overall, this project allows me to challenge myself consistently, applying computational thinking, and collaborate as a team efficiently.

This project would be more meaningful if it involves creating an application to tackle society/environmental issues.

Kalki

It was a challenging task as I was the main person of handing the data structures of the menus and other stores information. From the help of complementary lectures on file management and advanced functions gave me a better understanding on how to integrate the knowledge into the project.

The experience gained from this project it gave me in depth knowledge of exploring other libraries beyond what I learned from lectures and labs within the short period. Applying to my mini project and making sure that the code is readable by adding comments to my group mate to understand my code better. We can further improve by adding database like using MYSQL then text files for larger scale of data in the future.

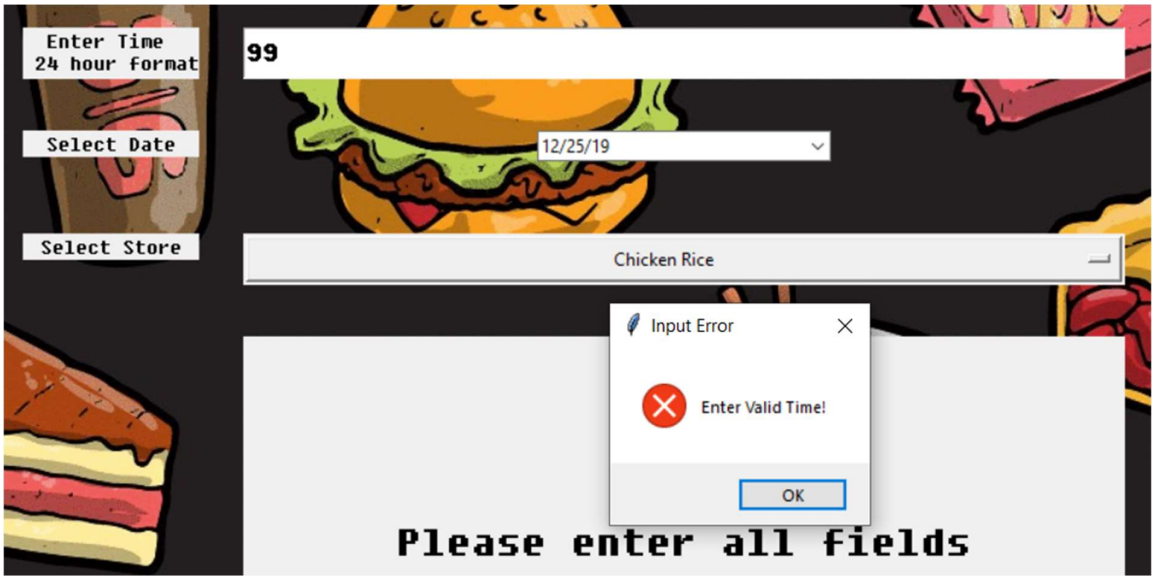
Owen

It was difficult to collaborate as a team and understand other people's code and make it work together. It is important for code to be modular and readable by adding comments and use appropriate variable names. Hard coding functions with many if else statements is not useful as the code is not reusable for other situations, but this was a feature we want to demonstrate with our menu system.

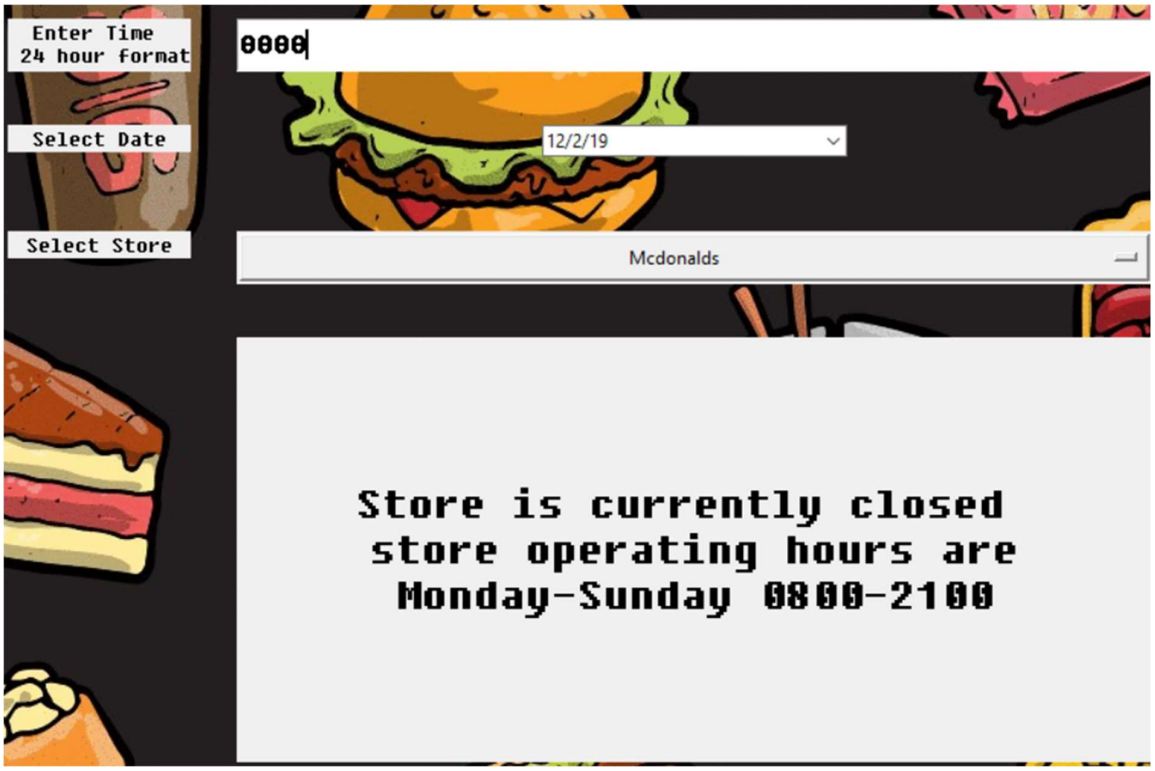
Using define functions, we are able to create and call other people's functions easily and incorporate it into our code, improving efficiency. We have settled on a file structure to store information with dictionary store(key) and menu(value) pairs as learnt in class to store information. However, manually Keying in menus of many stores in our dictionary is tedious and we can improve by using a database to fetch data easier and allow our menu system to be scalable in the future.

Appendix

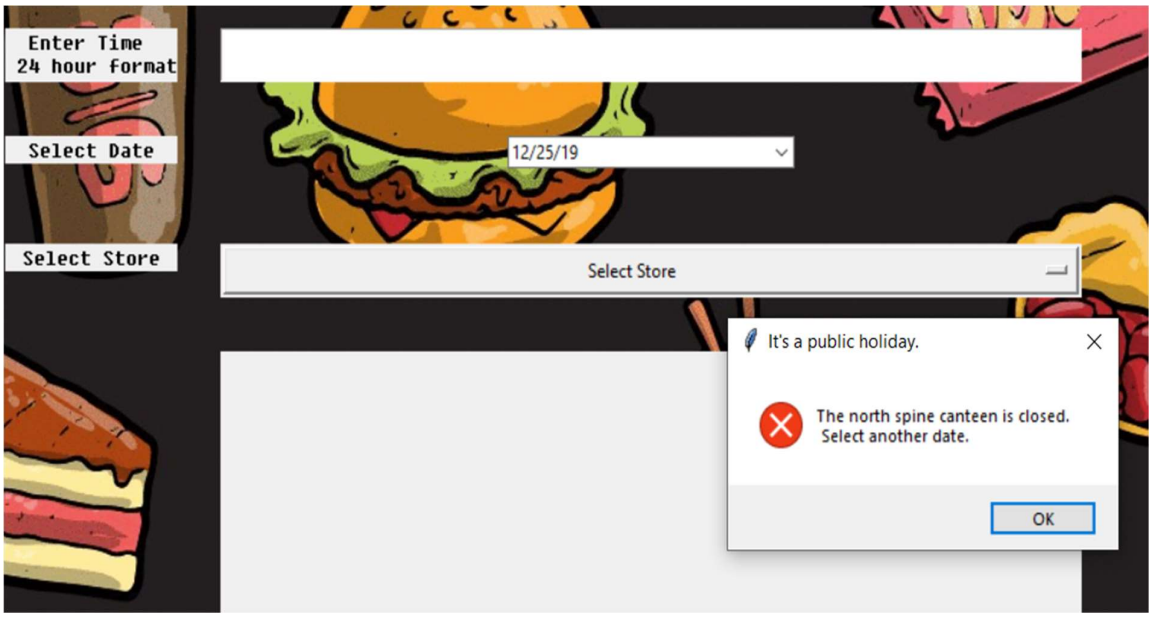
Appendix 1.1



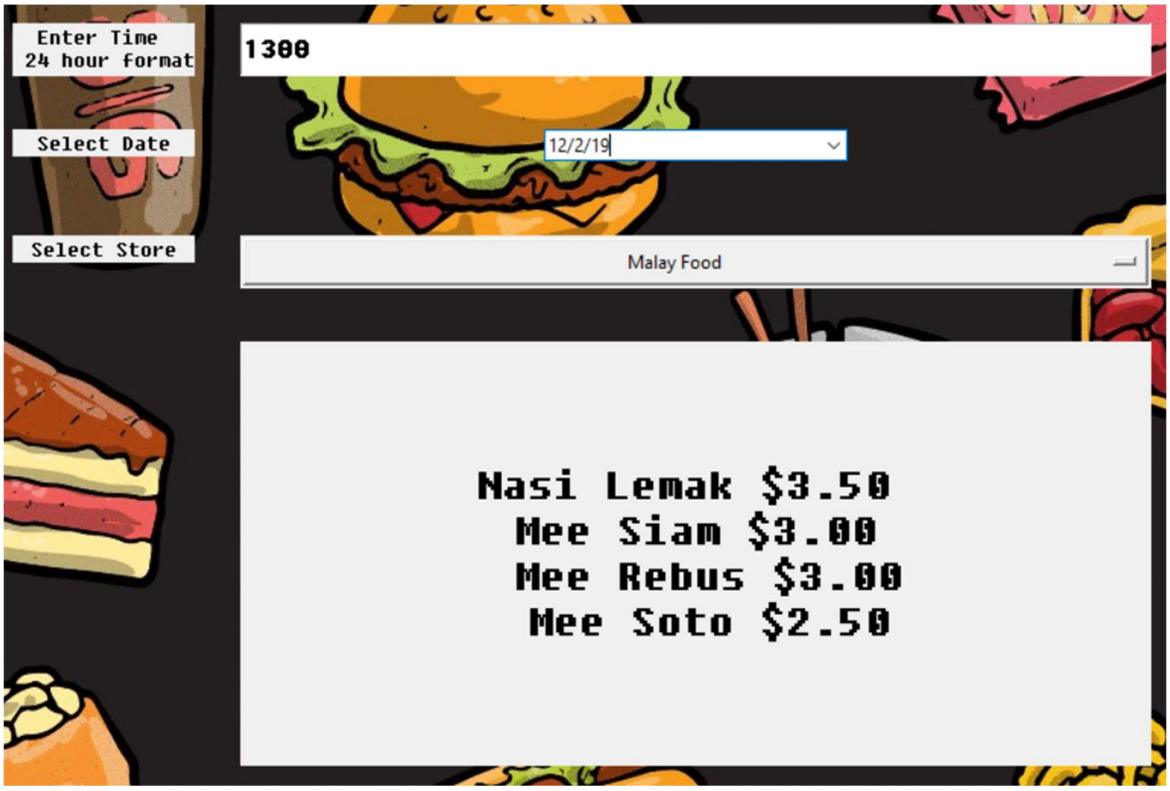
Appendix 1.2



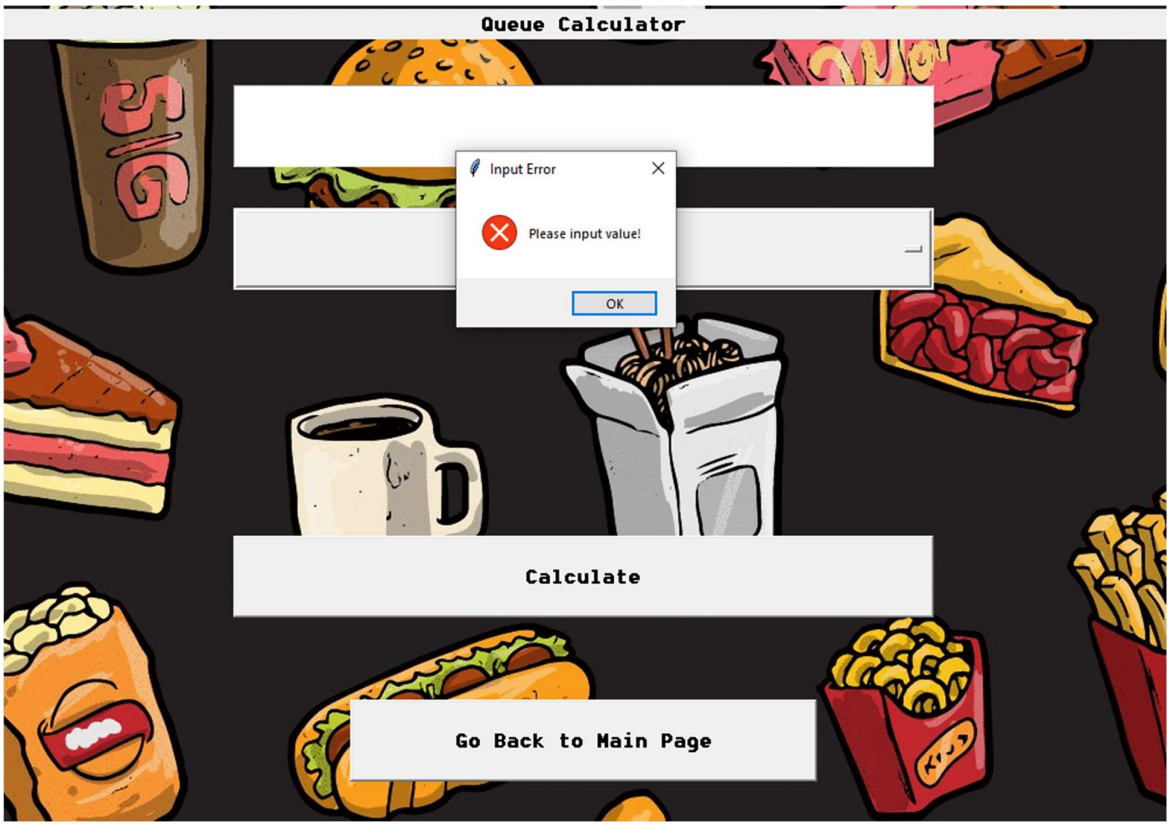
Appendix 1.3



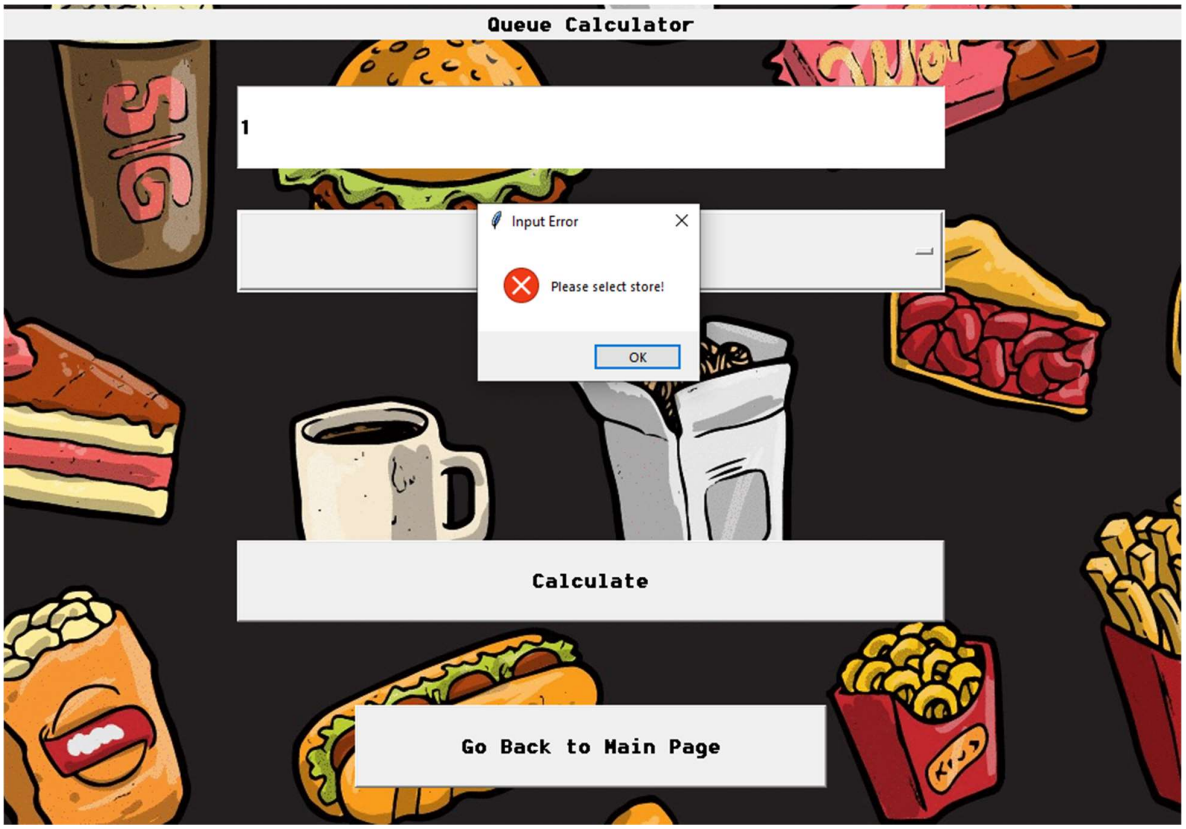
Appendix 1.4



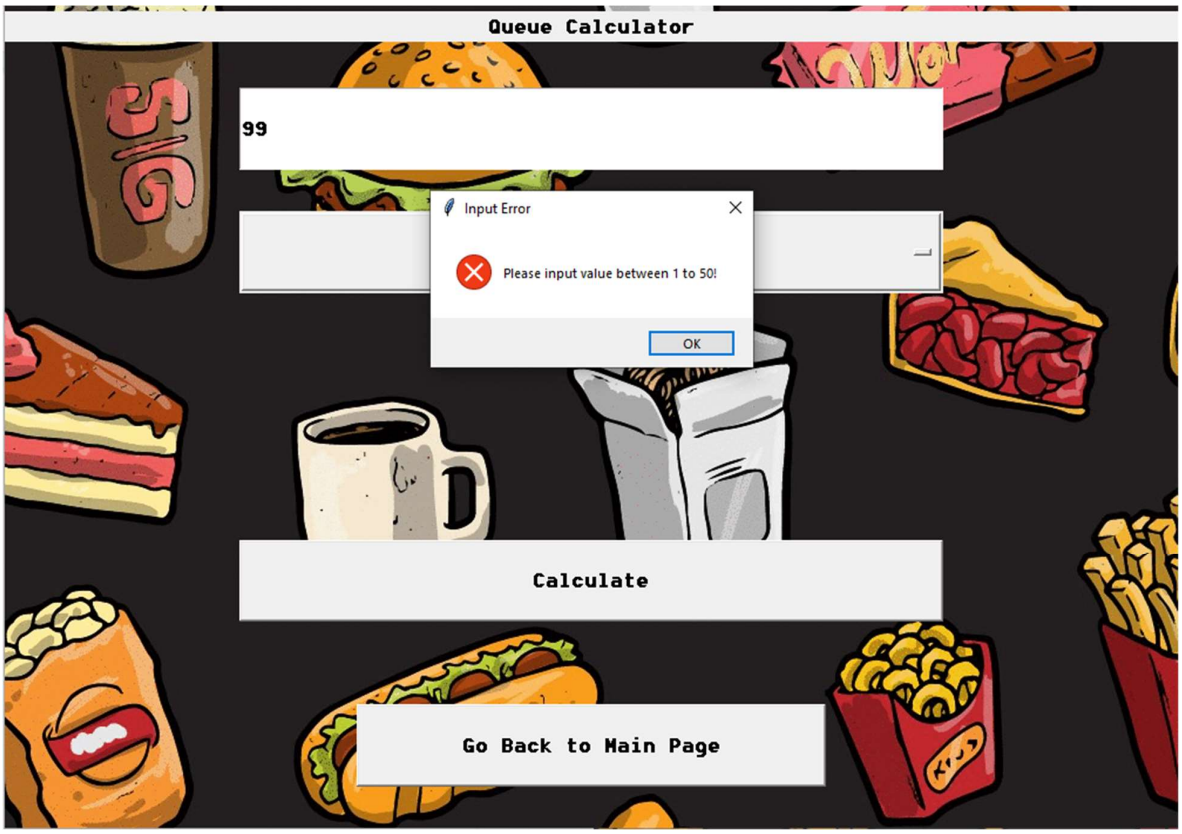
Appendix 1.5



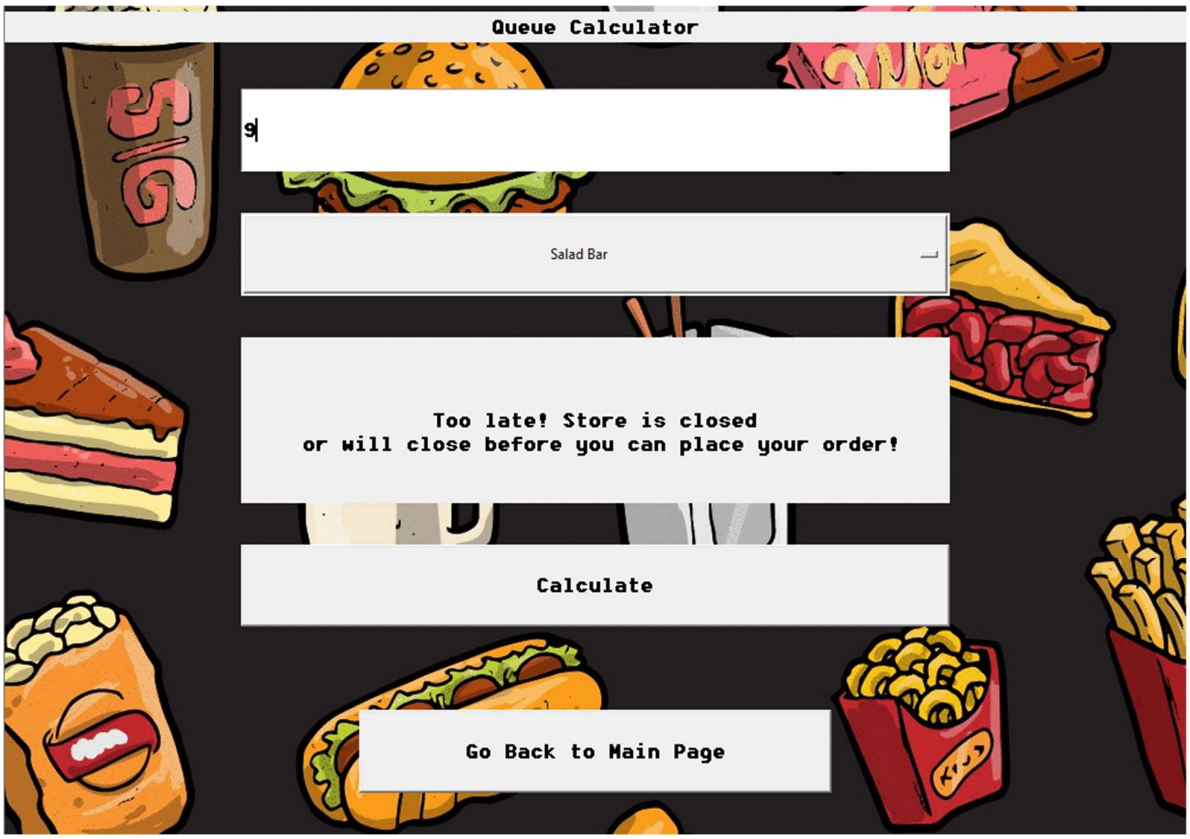
Appendix 2.1



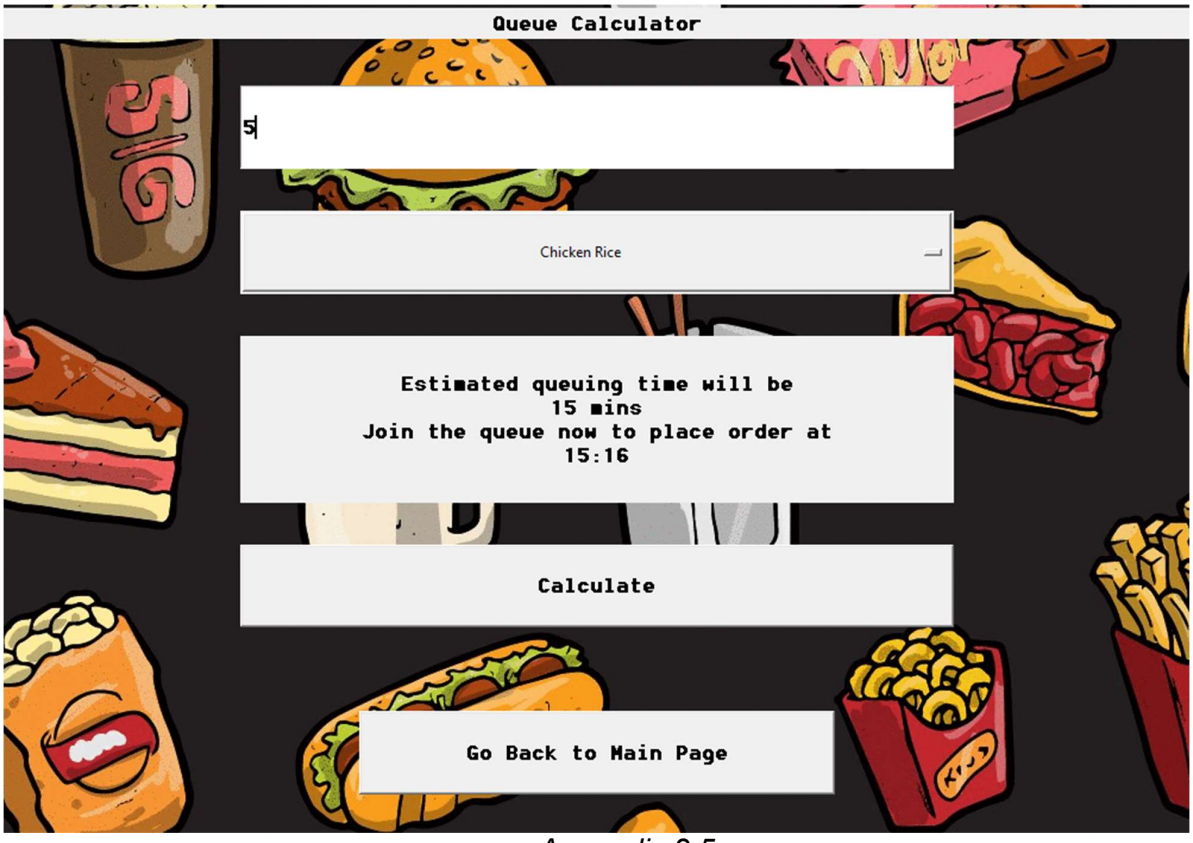
Appendix 2.2



Appendix 2.3



Appendix 2.4



Appendix 2.5