CE1003

Intro to Computational Thinking

Mini Project

Real-time Canteen Information System

Tutorial Group:	FE2
Member 1:	Lee Kai En
Matriculation No.:	U1922980A
Member 2:	Lee Jian Hao Andrew
Matriculation No.:	U1921895J

TABLE OF CONTENTS

Contributions of Each Member

Member 1: Lee Kai En*

- Feature C
- Feature E
- Program Compilation

Kai En contributed in fulfilling Feature C and E, retrieving the menu based on the time and date. If outside operating hours, the system will not proceed. Feature E is achieved through pop-up window to allow users to pick then the program will compute the results.

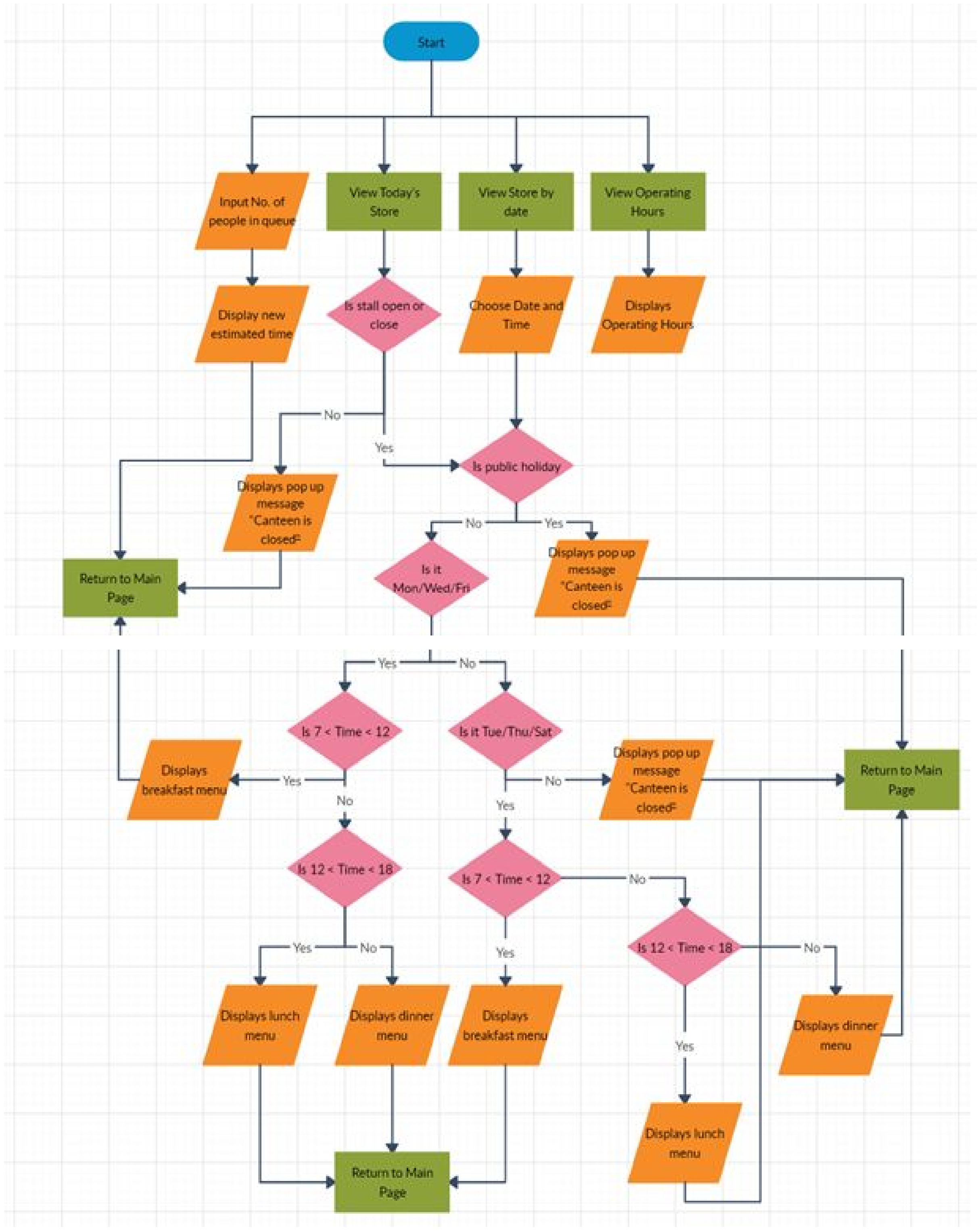
Member 2: Lee Jian Hao Andrew*

- Feature D
- Feature F

Andrew contributed in fulfilling Features D and F, allowing users to select their preferred date and time to view day-specific and time-specific menus through the use of a calendar for Feature D and calling a text file to display operating hours to fulfill Feature F.

*Both members worked on Features A & B together.

Algorithm Design
Top level flow-chart



Brief Description of Important User-Defined Functions

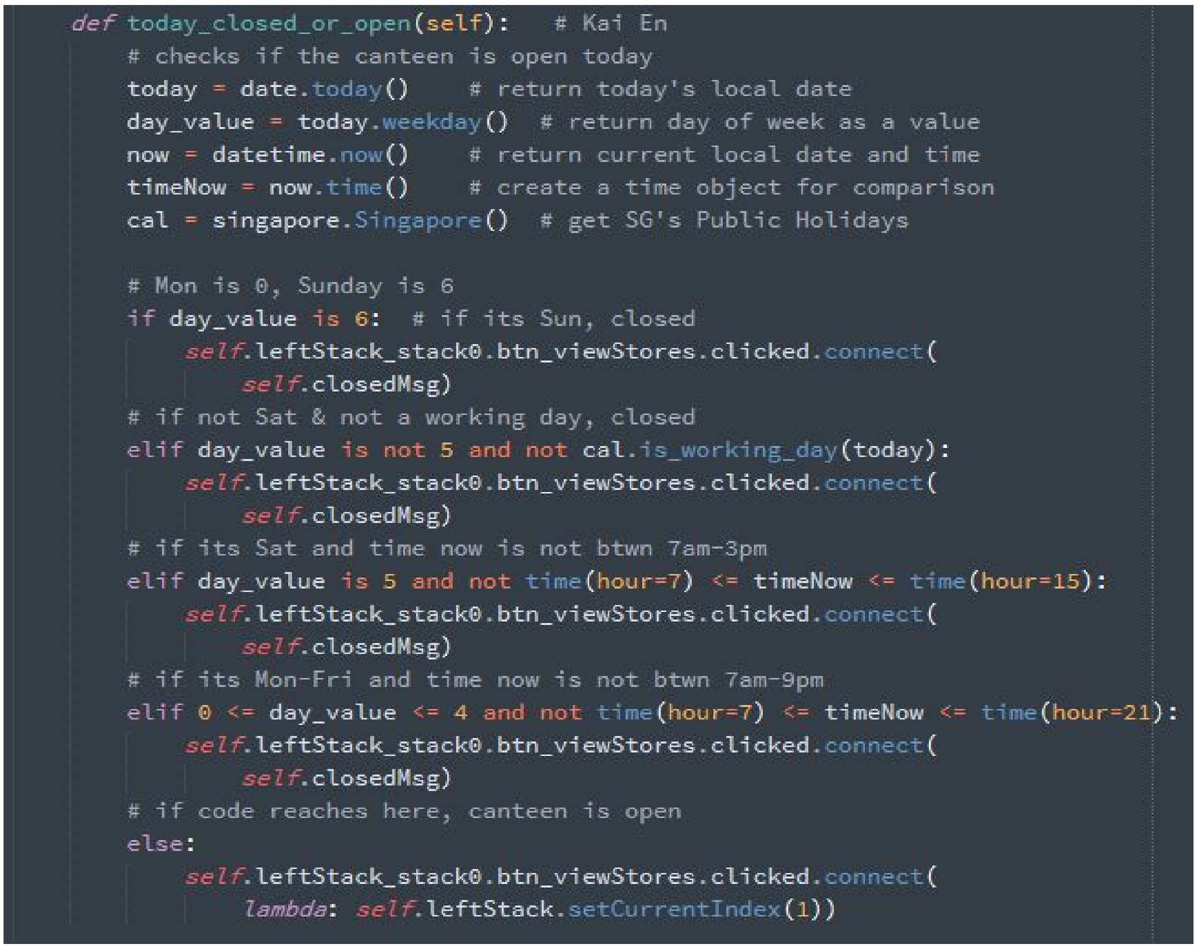
<u>Function Feature C:</u>

Display menu based on current system date & time

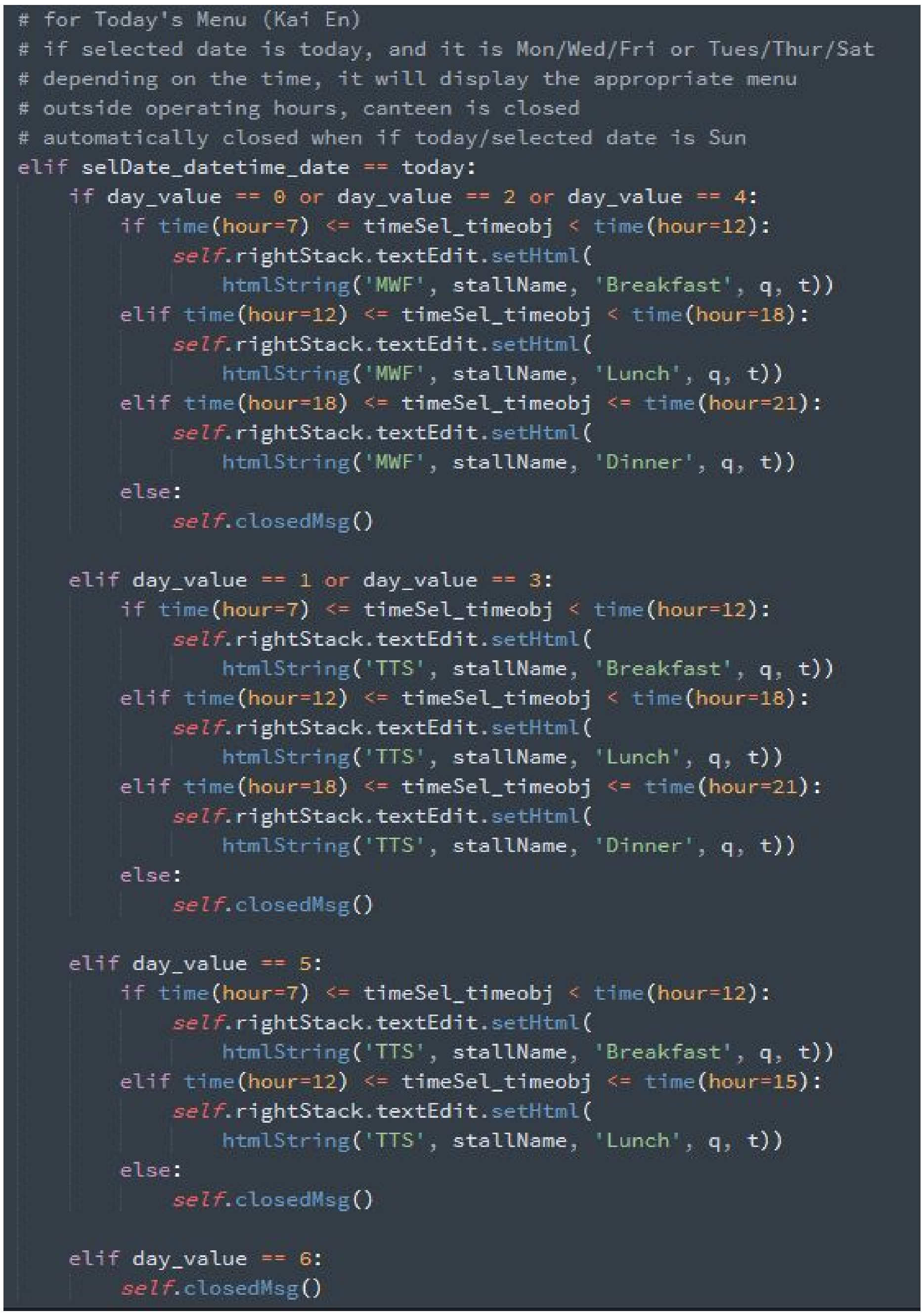
External Modules Used:

PyQt5, datetime, workalendar

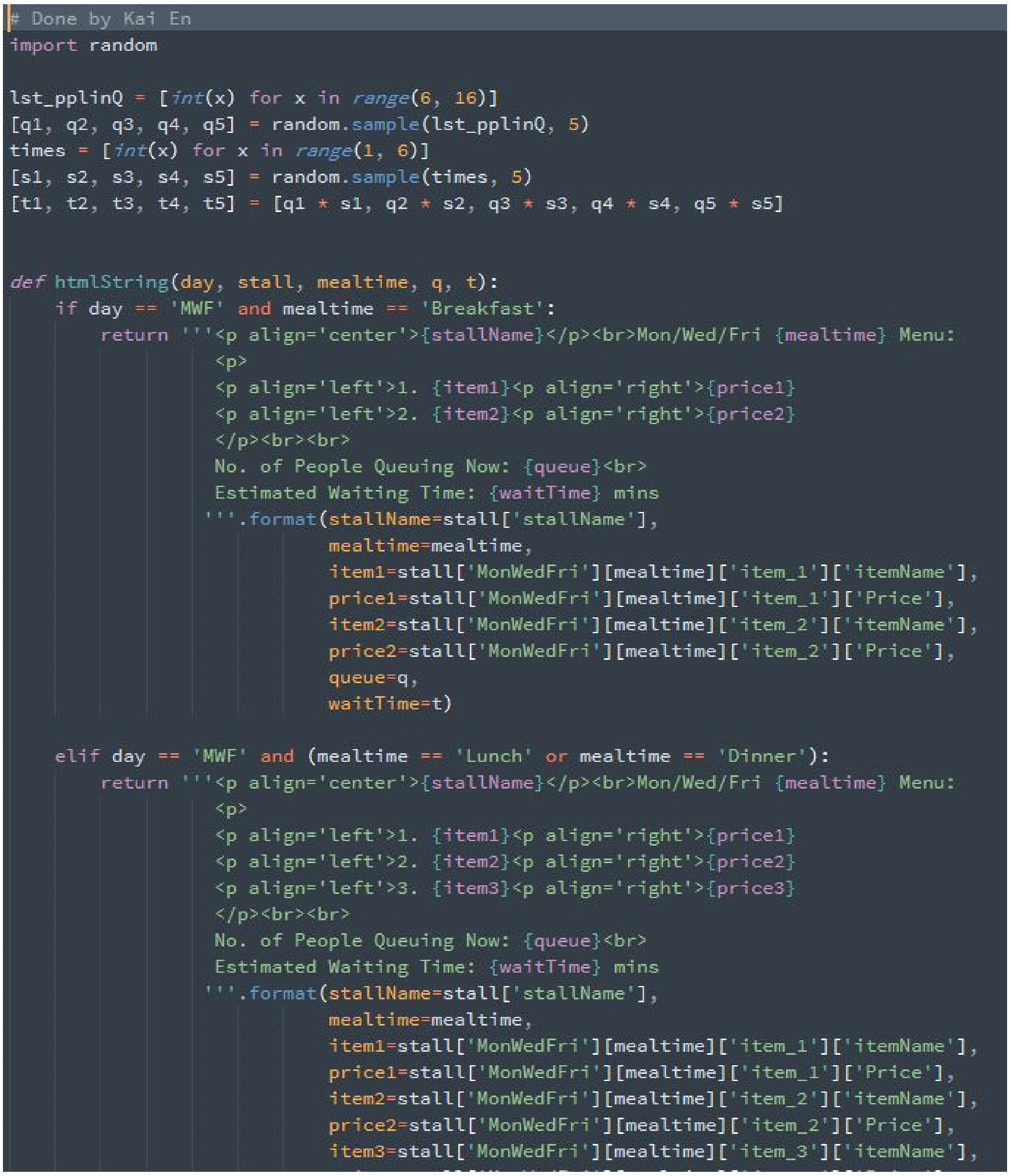
Code:



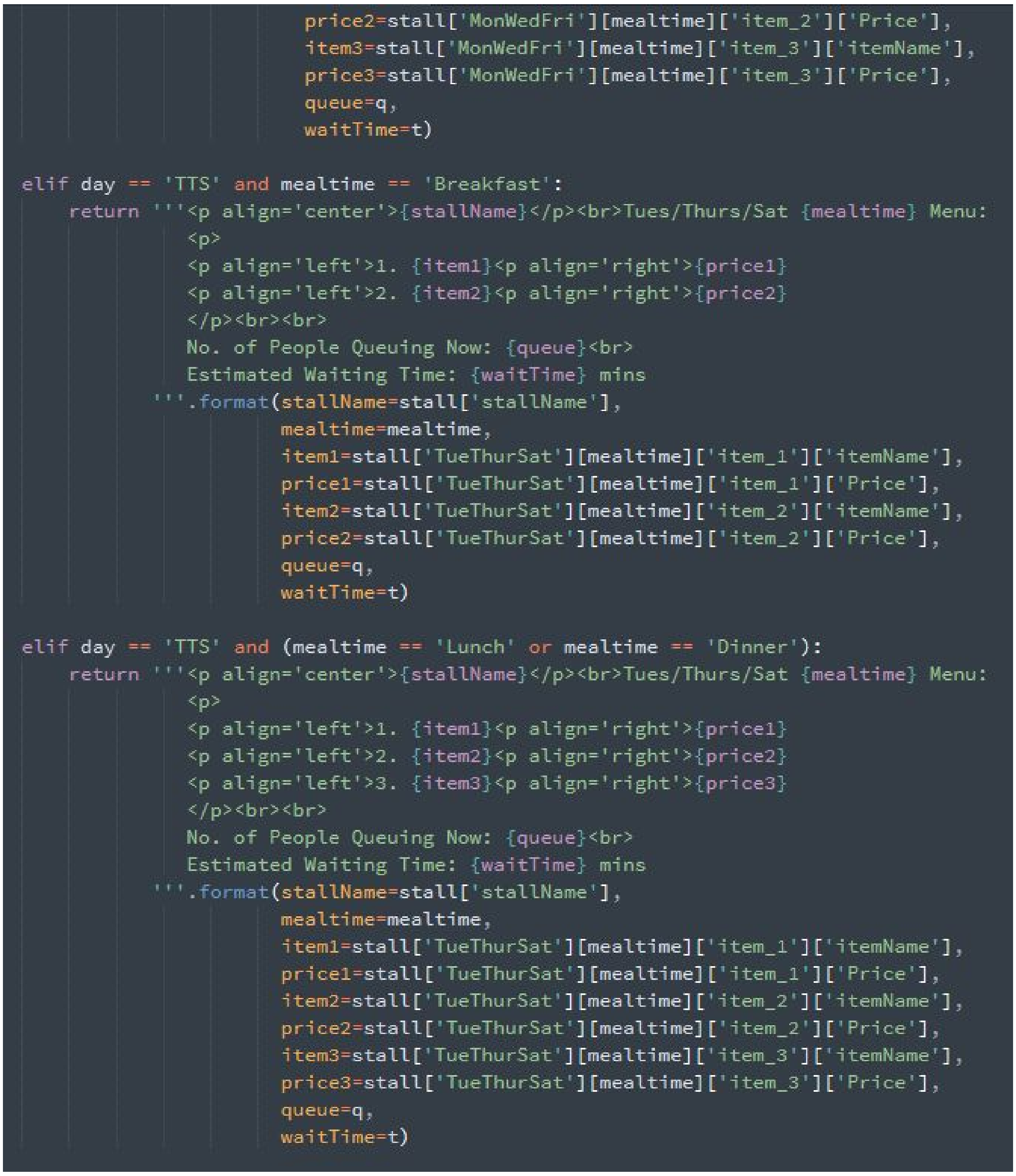
```
def today_closed_or_open(self):   # Kai En
    # checks if the canteen is open today
    today = date.today()    # return today's local date
    day_value = today.weekday()  # return day of week as a value
    now = datetime.now()    # return current local date and time
    timeNow = now.time()    # create a time object for comparison
    cal = singapore.Singapore()  # get SG's Public Holidays

    # Mon is 0, Sunday is 6
    if day_value is 6:  # if its Sun, closed
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if not Sat & not a working day, closed
    elif day_value is not 5 and not cal.is_working_day(today):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if its Sat and time now is not btwn 7am-3pm
    elif day_value is 5 and not time(hour=7) <= timeNow <= time(hour=15):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if its Mon-Fri and time now is not btwn 7am-9pm
    elif 0 <= day_value <= 4 and not time(hour=7) <= timeNow <= time(hour=21):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if code reaches here, canteen is open
    else:
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            lambda: self.leftStack.setCurrentIndex(1))
```



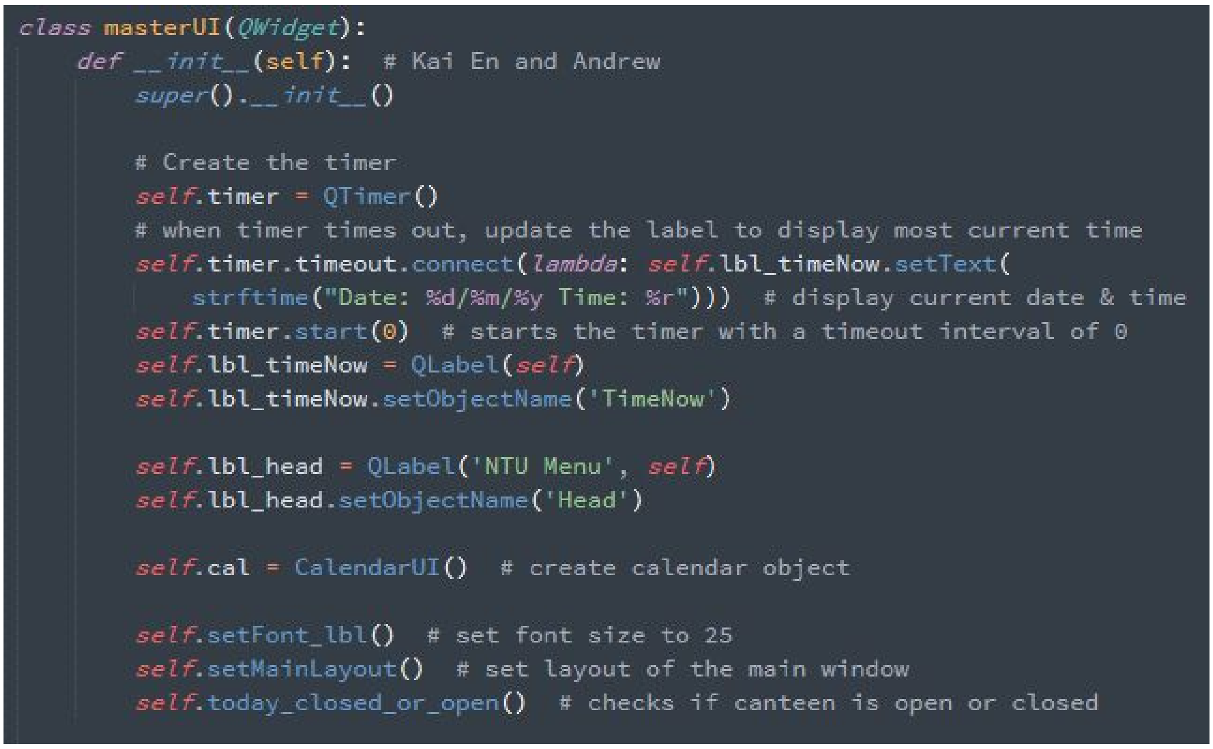
```python
# for Today's Menu (Kai En)
# if selected date is today, and it is Mon/Wed/Fri or Tues/Thur/Sat
# depending on the time, it will display the appropriate menu
# outside operating hours, canteen is closed
# automatically closed when if today/selected date is Sun
elif selDate_datetime_date == today:
    if day_value == 0 or day_value == 2 or day_value == 4:
        if time(hour=7) <= timeSel_timeobj < time(hour=12):
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Breakfast', q, t))
        elif time(hour=12) <= timeSel_timeobj < time(hour=18):
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Lunch', q, t))
        elif time(hour=18) <= timeSel_timeobj <= time(hour=21):
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Dinner', q, t))
        else:
            self.closedMsg()

    elif day_value == 1 or day_value == 3:
        if time(hour=7) <= timeSel_timeobj < time(hour=12):
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Breakfast', q, t))
        elif time(hour=12) <= timeSel_timeobj < time(hour=18):
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Lunch', q, t))
        elif time(hour=18) <= timeSel_timeobj <= time(hour=21):
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Dinner', q, t))
        else:
            self.closedMsg()

    elif day_value == 5:
        if time(hour=7) <= timeSel_timeobj < time(hour=12):
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Breakfast', q, t))
        elif time(hour=12) <= timeSel_timeobj <= time(hour=15):
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Lunch', q, t))
        else:
            self.closedMsg()

    elif day_value == 6:
        self.closedMsg()
```



```python
# Done by Kai En
import random

lst_pplinQ = [int(x) for x in range(6, 16)]
[q1, q2, q3, q4, q5] = random.sample(lst_pplinQ, 5)
times = [int(x) for x in range(1, 6)]
[s1, s2, s3, s4, s5] = random.sample(times, 5)
[t1, t2, t3, t4, t5] = [q1 * s1, q2 * s2, q3 * s3, q4 * s4, q5 * s5]


def htmlString(day, stall, mealtime, q, t):
    if day == 'MWF' and mealtime == 'Breakfast':
        return '''<p align='center'>{stallName}</p><br>Mon/Wed/Fri {mealtime} Menu:
                  <p>
                  <p align='left'>1. {item1}<p align='right'>{price1}
                  <p align='left'>2. {item2}<p align='right'>{price2}
                  </p><br><br>
                  No. of People Queuing Now: {queue}<br>
                  Estimated Waiting Time: {waitTime} mins
                 '''.format(stallName=stall['stallName'],
                            mealtime=mealtime,
                            item1=stall['MonWedFri'][mealtime]['item_1']['itemName'],
                            price1=stall['MonWedFri'][mealtime]['item_1']['Price'],
                            item2=stall['MonWedFri'][mealtime]['item_2']['itemName'],
                            price2=stall['MonWedFri'][mealtime]['item_2']['Price'],
                            queue=q,
                            waitTime=t)

    elif day == 'MWF' and (mealtime == 'Lunch' or mealtime == 'Dinner'):
        return '''<p align='center'>{stallName}</p><br>Mon/Wed/Fri {mealtime} Menu:
                  <p>
                  <p align='left'>1. {item1}<p align='right'>{price1}
                  <p align='left'>2. {item2}<p align='right'>{price2}
                  <p align='left'>3. {item3}<p align='right'>{price3}
                  </p><br><br>
                  No. of People Queuing Now: {queue}<br>
                  Estimated Waiting Time: {waitTime} mins
                 '''.format(stallName=stall['stallName'],
                            mealtime=mealtime,
                            item1=stall['MonWedFri'][mealtime]['item_1']['itemName'],
                            price1=stall['MonWedFri'][mealtime]['item_1']['Price'],
                            item2=stall['MonWedFri'][mealtime]['item_2']['itemName'],
                            price2=stall['MonWedFri'][mealtime]['item_2']['Price'],
                            item3=stall['MonWedFri'][mealtime]['item_3']['itemName'],
```



```
                        price2=stall['MonWedFri'][mealtime]['item_2']['Price'],
                        item3=stall['MonWedFri'][mealtime]['item_3']['itemName'],
                        price3=stall['MonWedFri'][mealtime]['item_3']['Price'],
                        queue=q,
                        waitTime=t)

elif day == 'TTS' and mealtime == 'Breakfast':
    return '''<p align='center'>{stallName}</p><br>Tues/Thurs/Sat {mealtime} Menu:
              <p>
              <p align='left'>1. {item1}<p align='right'>{price1}
              <p align='left'>2. {item2}<p align='right'>{price2}
              </p><br><br>
              No. of People Queuing Now: {queue}<br>
              Estimated Waiting Time: {waitTime} mins
           '''.format(stallName=stall['stallName'],
                      mealtime=mealtime,
                      item1=stall['TueThurSat'][mealtime]['item_1']['itemName'],
                      price1=stall['TueThurSat'][mealtime]['item_1']['Price'],
                      item2=stall['TueThurSat'][mealtime]['item_2']['itemName'],
                      price2=stall['TueThurSat'][mealtime]['item_2']['Price'],
                      queue=q,
                      waitTime=t)

elif day == 'TTS' and (mealtime == 'Lunch' or mealtime == 'Dinner'):
    return '''<p align='center'>{stallName}</p><br>Tues/Thurs/Sat {mealtime} Menu:
              <p>
              <p align='left'>1. {item1}<p align='right'>{price1}
              <p align='left'>2. {item2}<p align='right'>{price2}
              <p align='left'>3. {item3}<p align='right'>{price3}
              </p><br><br>
              No. of People Queuing Now: {queue}<br>
              Estimated Waiting Time: {waitTime} mins
           '''.format(stallName=stall['stallName'],
                      mealtime=mealtime,
                      item1=stall['TueThurSat'][mealtime]['item_1']['itemName'],
                      price1=stall['TueThurSat'][mealtime]['item_1']['Price'],
                      item2=stall['TueThurSat'][mealtime]['item_2']['itemName'],
                      price2=stall['TueThurSat'][mealtime]['item_2']['Price'],
                      item3=stall['TueThurSat'][mealtime]['item_3']['itemName'],
                      price3=stall['TueThurSat'][mealtime]['item_3']['Price'],
                      queue=q,
                      waitTime=t)
```



```python
class masterUI(QWidget):
    def __init__(self):  # Kai En and Andrew
        super().__init__()

        # Create the timer
        self.timer = QTimer()
        # when timer times out, update the label to display most current time
        self.timer.timeout.connect(lambda: self.lbl_timeNow.setText(
            strftime("Date: %d/%m/%y Time: %r")))  # display current date & time
        self.timer.start(0)  # starts the timer with a timeout interval of 0
        self.lbl_timeNow = QLabel(self)
        self.lbl_timeNow.setObjectName('TimeNow')

        self.lbl_head = QLabel('NTU Menu', self)
        self.lbl_head.setObjectName('Head')

        self.cal = CalendarUI()  # create calendar object

        self.setFont_lbl()  # set font size to 25
        self.setMainLayout()  # set layout of the main window
        self.today_closed_or_open()  # checks if canteen is open or closed
```

Description:

Firstly, the program checks if the canteen is open or closed today by comparing against the operating hours and public holiday dates. If closed, it will not proceed onto the next UI. If open, clicking the ‘View Stores’ Button will redirect user to select the desired stall. Since the calendar’s selected date is today’s date by default, this function is designed to encompass both today’s date and the calendar’s selected date to reduce the need for another similarly-defined function.

The program then checks if it is Monday, Wednesday, Friday or Tuesday, Thursday, Saturday and checks if breakfast (7am-12nn), lunch (12nn-6pm) or dinner (6pm-9pm) before proceeding to display the corresponding menu. Outside the above-mentioned hours, the canteen is closed and a ‘Closed’ message will appear.

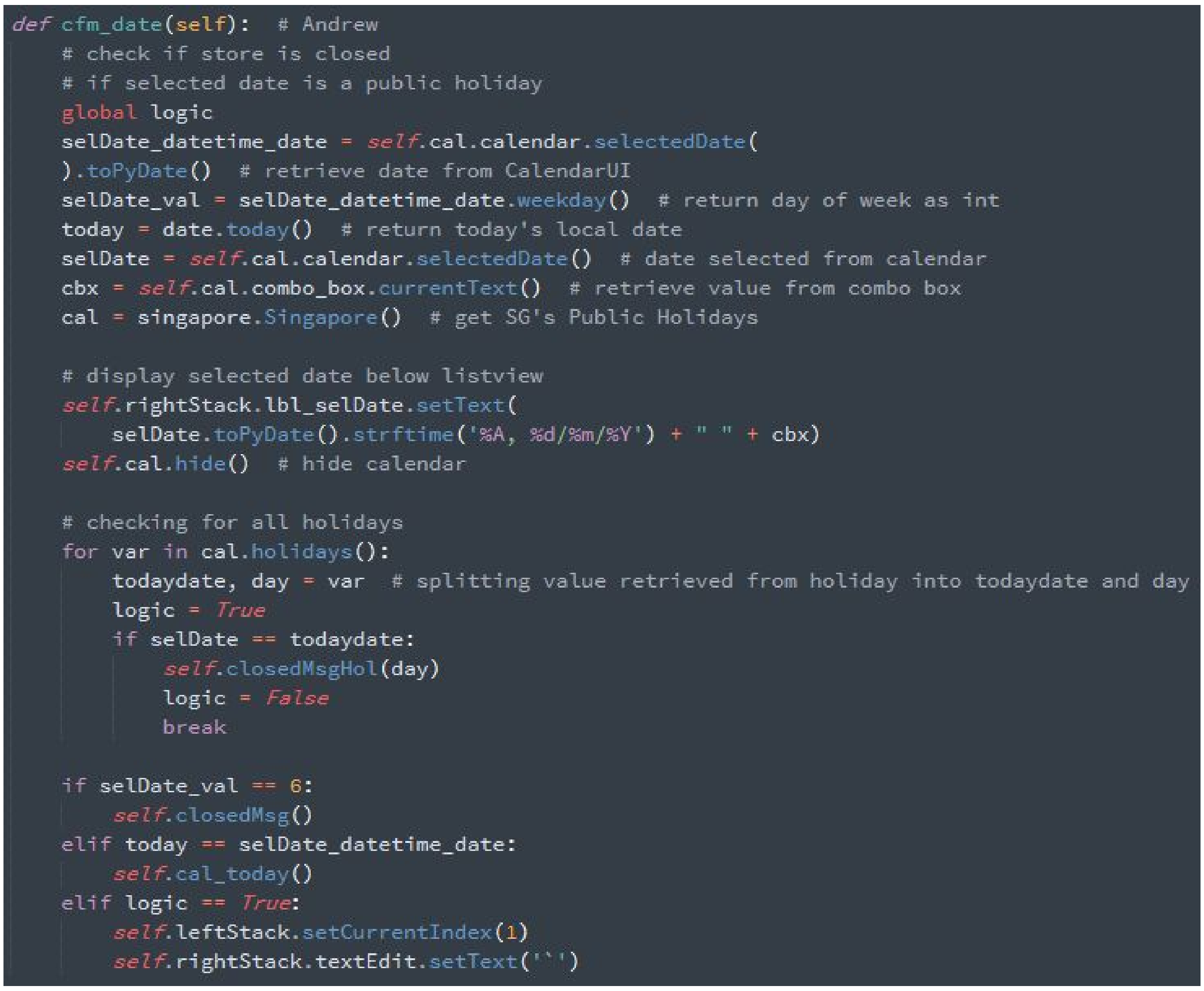
<u>Function Feature D:</u>

Display menu based on user defined system date & time

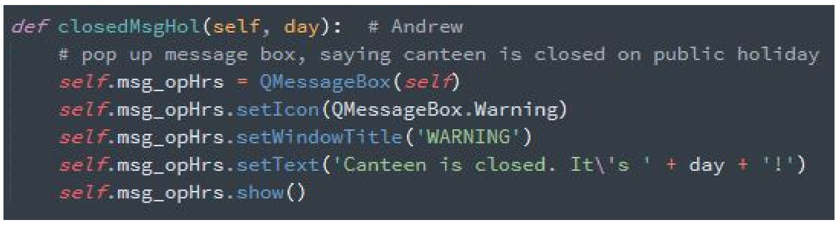
External Modules Used:

PyQt5, datetime, workalendar

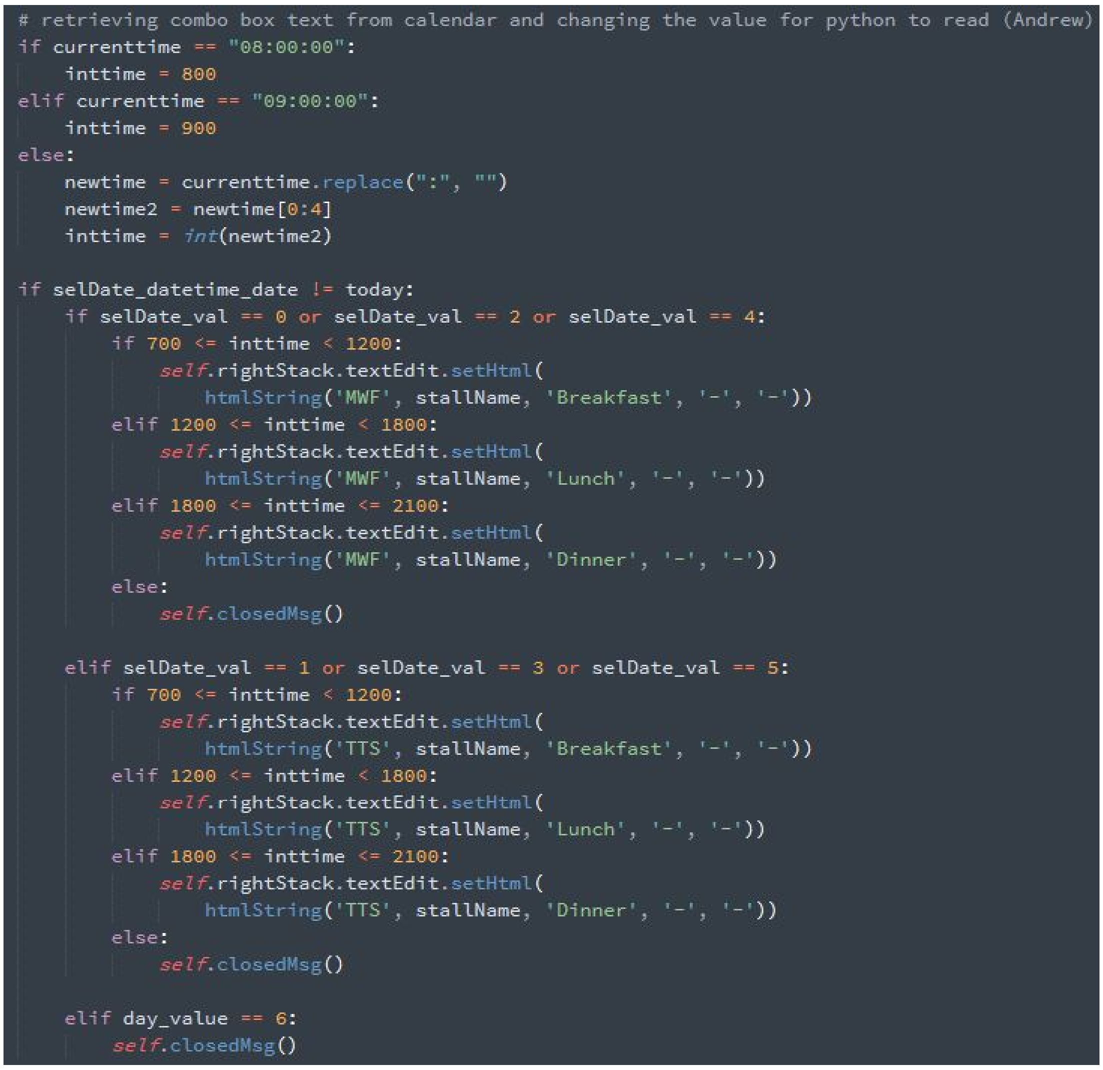
Code:



```python
def cfm_date(self):  # Andrew
    # check if store is closed
    # if selected date is a public holiday
    global logic
    selDate_datetime_date = self.cal.calendar.selectedDate(
    ).toPyDate()  # retrieve date from CalendarUI
    selDate_val = selDate_datetime_date.weekday()  # return day of week as int
    today = date.today()  # return today's local date
    selDate = self.cal.calendar.selectedDate()  # date selected from calendar
    cbx = self.cal.combo_box.currentText()  # retrieve value from combo box
    cal = singapore.Singapore()  # get SG's Public Holidays

    # display selected date below listview
    self.rightStack.lbl_selDate.setText(
        selDate.toPyDate().strftime('%A, %d/%m/%Y') + " " + cbx)
    self.cal.hide()  # hide calendar

    # checking for all holidays
    for var in cal.holidays():
        todaydate, day = var  # splitting value retrieved from holiday into todaydate and day
        logic = True
        if selDate == todaydate:
            self.closedMsgHol(day)
            logic = False
            break

    if selDate_val == 6:
        self.closedMsg()
    elif today == selDate_datetime_date:
        self.cal_today()
    elif logic == True:
        self.leftStack.setCurrentIndex(1)
        self.rightStack.textEdit.setText('`')
```



```python
def closedMsgHol(self, day):  # Andrew
    # pop up message box, saying canteen is closed on public holiday
    self.msg_opHrs = QMessageBox(self)
    self.msg_opHrs.setIcon(QMessageBox.Warning)
    self.msg_opHrs.setWindowTitle('WARNING')
    self.msg_opHrs.setText('Canteen is closed. It\'s ' + day + '!')
    self.msg_opHrs.show()
```



```python
# retrieving combo box text from calendar and changing the value for python to read (Andrew)
if currenttime == "08:00:00":
    inttime = 800
elif currenttime == "09:00:00":
    inttime = 900
else:
    newtime = currenttime.replace(":", "")
    newtime2 = newtime[0:4]
    inttime = int(newtime2)

if selDate_datetime_date != today:
    if selDate_val == 0 or selDate_val == 2 or selDate_val == 4:
        if 700 <= inttime < 1200:
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Breakfast', '-', '-'))
        elif 1200 <= inttime < 1800:
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Lunch', '-', '-'))
        elif 1800 <= inttime <= 2100:
            self.rightStack.textEdit.setHtml(
                htmlString('MWF', stallName, 'Dinner', '-', '-'))
        else:
            self.closedMsg()

    elif selDate_val == 1 or selDate_val == 3 or selDate_val == 5:
        if 700 <= inttime < 1200:
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Breakfast', '-', '-'))
        elif 1200 <= inttime < 1800:
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Lunch', '-', '-'))
        elif 1800 <= inttime <= 2100:
            self.rightStack.textEdit.setHtml(
                htmlString('TTS', stallName, 'Dinner', '-', '-'))
        else:
            self.closedMsg()

    elif day_value == 6:
        self.closedMsg()
```

Description:

For the first part, the program will check if the user-selected date falls on a holiday and if it does, a 'Closed' message will appear after confirming the selection. Otherwise, the store buttons will appear for selection to set the menu on display. Next, the program takes the time selected by user and converts it into an integer. If selected date is not today, the if-else statements will continue to check which day group it belongs, whether it is Monday, Wednesday, Friday or Tuesday, Thursday, Saturday. Next, it checks the time range the user-selected time falls into and sets the corresponding menu on display. If the selected time is not within the operating hours, a 'Closed' message will appear.

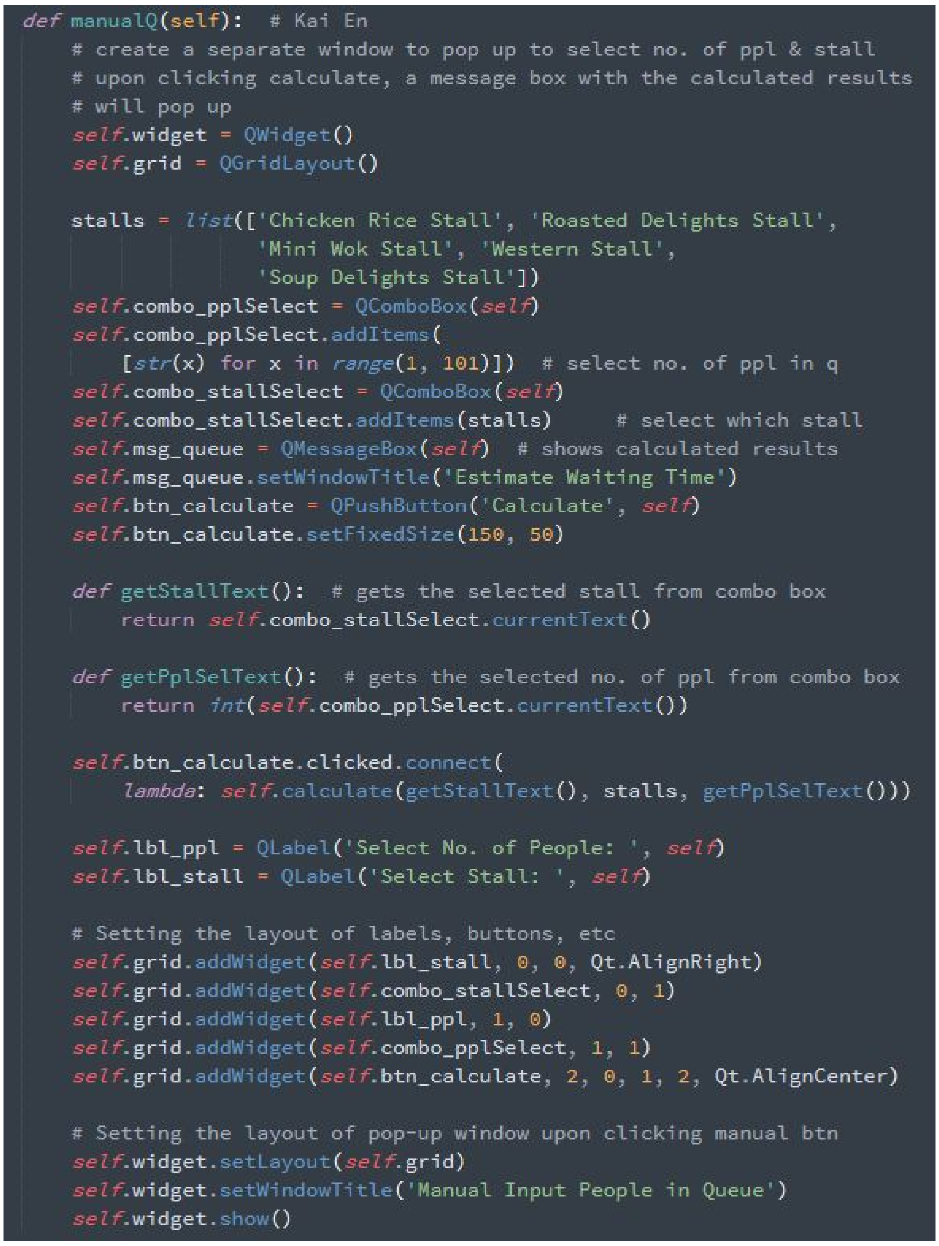
Function Feature E:

Manually input the number of people in the queue to calculate a new estimated waiting time

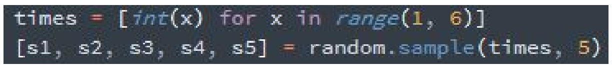
External Modules Used:

PyQt5, random

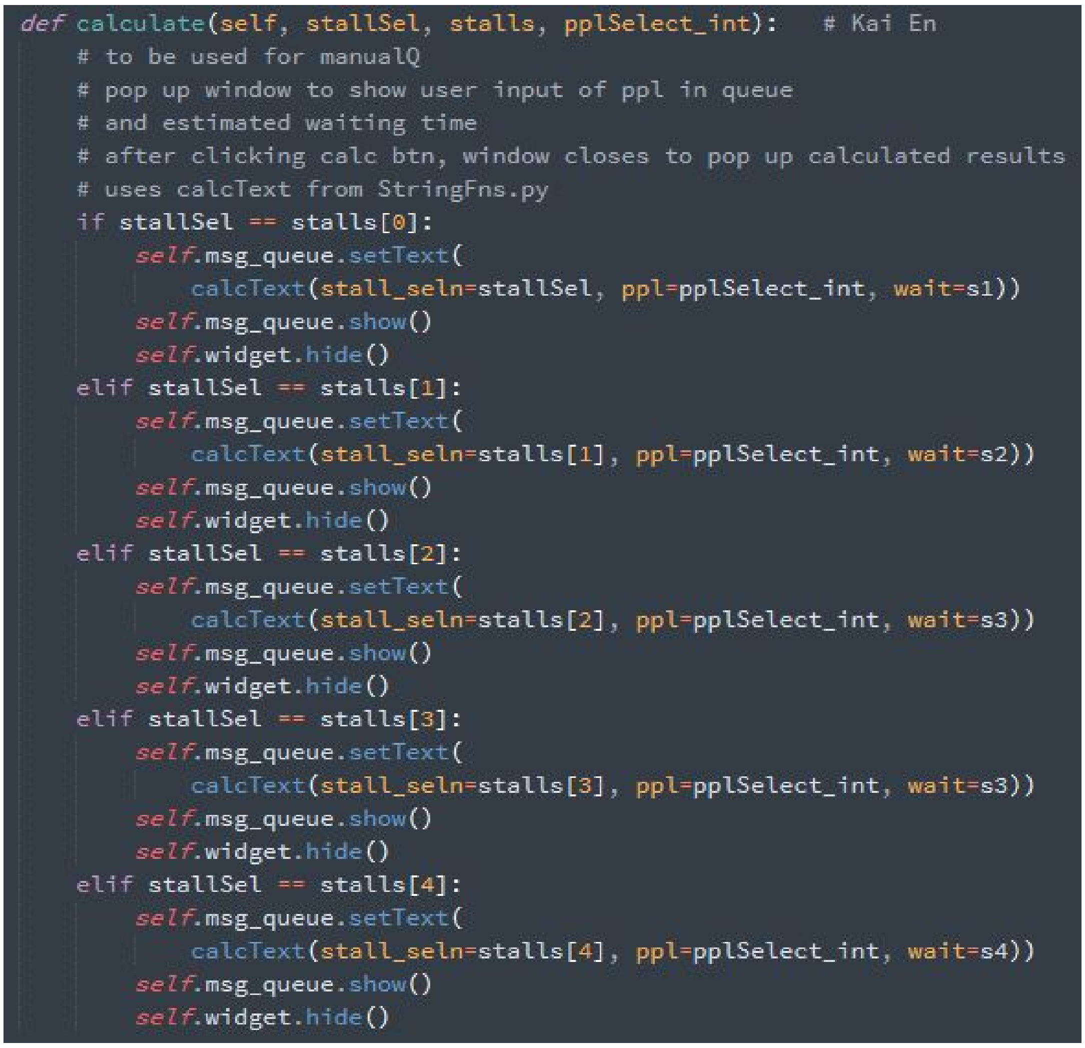
Code:



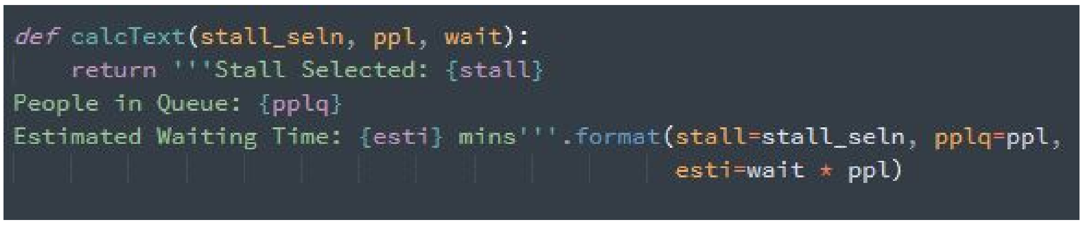
```
def manualQ(self):  # Kai En
    # create a separate window to pop up to select no. of ppl & stall
    # upon clicking calculate, a message box with the calculated results
    # will pop up
    self.widget = QWidget()
    self.grid = QGridLayout()

    stalls = list(['Chicken Rice Stall', 'Roasted Delights Stall',
                   'Mini Wok Stall', 'Western Stall',
                   'Soup Delights Stall'])
    self.combo_pplSelect = QComboBox(self)
    self.combo_pplSelect.addItems(
        [str(x) for x in range(1, 101)])  # select no. of ppl in q
    self.combo_stallSelect = QComboBox(self)
    self.combo_stallSelect.addItems(stalls)     # select which stall
    self.msg_queue = QMessageBox(self)  # shows calculated results
    self.msg_queue.setWindowTitle('Estimate Waiting Time')
    self.btn_calculate = QPushButton('Calculate', self)
    self.btn_calculate.setFixedSize(150, 50)

    def getStallText():  # gets the selected stall from combo box
        return self.combo_stallSelect.currentText()

    def getPplSelText():  # gets the selected no. of ppl from combo box
        return int(self.combo_pplSelect.currentText())

    self.btn_calculate.clicked.connect(
        lambda: self.calculate(getStallText(), stalls, getPplSelText()))

    self.lbl_ppl = QLabel('Select No. of People: ', self)
    self.lbl_stall = QLabel('Select Stall: ', self)

    # Setting the layout of labels, buttons, etc
    self.grid.addWidget(self.lbl_stall, 0, 0, Qt.AlignRight)
    self.grid.addWidget(self.combo_stallSelect, 0, 1)
    self.grid.addWidget(self.lbl_ppl, 1, 0)
    self.grid.addWidget(self.combo_pplSelect, 1, 1)
    self.grid.addWidget(self.btn_calculate, 2, 0, 1, 2, Qt.AlignCenter)

    # Setting the layout of pop-up window upon clicking manual btn
    self.widget.setLayout(self.grid)
    self.widget.setWindowTitle('Manual Input People in Queue')
    self.widget.show()
```



```
times = [int(x) for x in range(1, 6)]
[s1, s2, s3, s4, s5] = random.sample(times, 5)
```



```
def calculate(self, stallSel, stalls, pplSelect_int):   # Kai En
    # to be used for manualQ
    # pop up window to show user input of ppl in queue
    # and estimated waiting time
    # after clicking calc btn, window closes to pop up calculated results
    # uses calcText from StringFns.py
    if stallSel == stalls[0]:
        self.msg_queue.setText(
            calcText(stall_seln=stallSel, ppl=pplSelect_int, wait=s1))
        self.msg_queue.show()
        self.widget.hide()
    elif stallSel == stalls[1]:
        self.msg_queue.setText(
            calcText(stall_seln=stalls[1], ppl=pplSelect_int, wait=s2))
        self.msg_queue.show()
        self.widget.hide()
    elif stallSel == stalls[2]:
        self.msg_queue.setText(
            calcText(stall_seln=stalls[2], ppl=pplSelect_int, wait=s3))
        self.msg_queue.show()
        self.widget.hide()
    elif stallSel == stalls[3]:
        self.msg_queue.setText(
            calcText(stall_seln=stalls[3], ppl=pplSelect_int, wait=s3))
        self.msg_queue.show()
        self.widget.hide()
    elif stallSel == stalls[4]:
        self.msg_queue.setText(
            calcText(stall_seln=stalls[4], ppl=pplSelect_int, wait=s4))
        self.msg_queue.show()
        self.widget.hide()
```



```
def calcText(stall_seln, ppl, wait):
    return '''Stall Selected: {stall}
People in Queue: {pplq}
Estimated Waiting Time: {esti} mins'''.format(stall=stall_seln, pplq=ppl,
                                              esti=wait * ppl)
```

Description:

Upon pressing the ‘Manual Input Q’ button to calculate the estimated waiting time, a new window will appear with combo boxes which allows the user to choose the number of people queuing, with possible values ranging from 1 to 100, and the desired stall. Then, the estimated waiting time will be calculated based on the stall selected and each has a randomized waiting time. Upon pressing the ‘Calculate’ button, the current window will close and a message box containing the calculated estimated waiting time for the stall will appear.

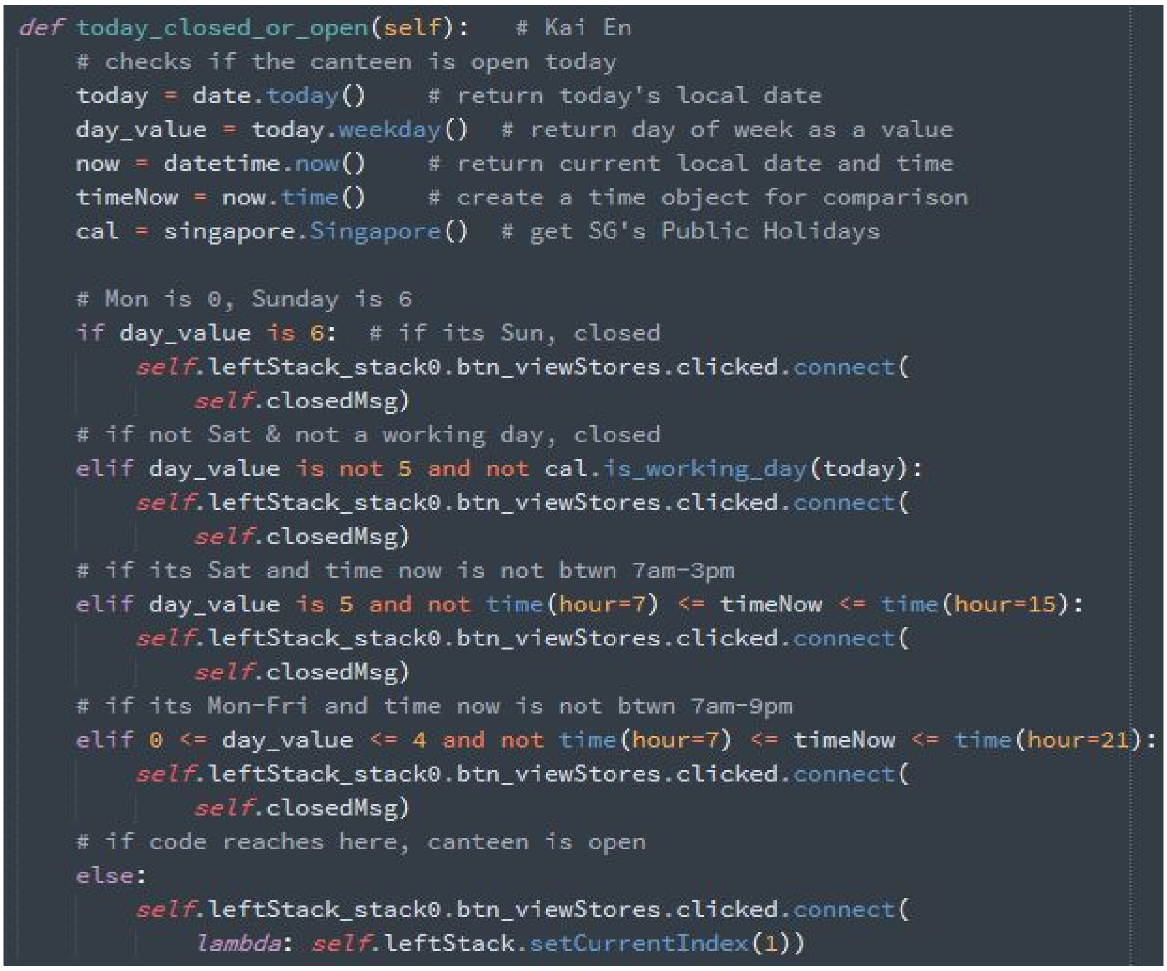
<u>Function Feature F:</u>

Check the operating hours of the stalls and enabling the program only when the canteen is open

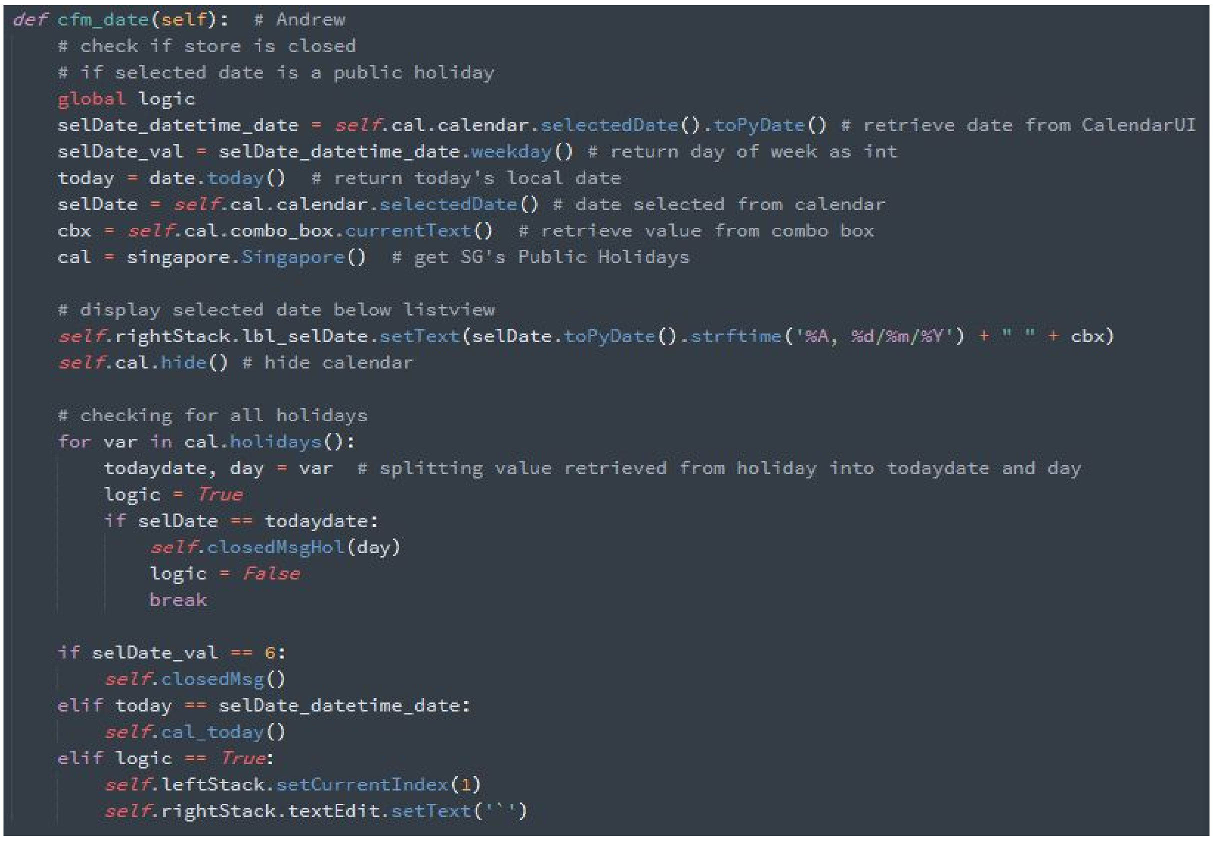
Modules Used:

PyQt5, datetime, workalendar

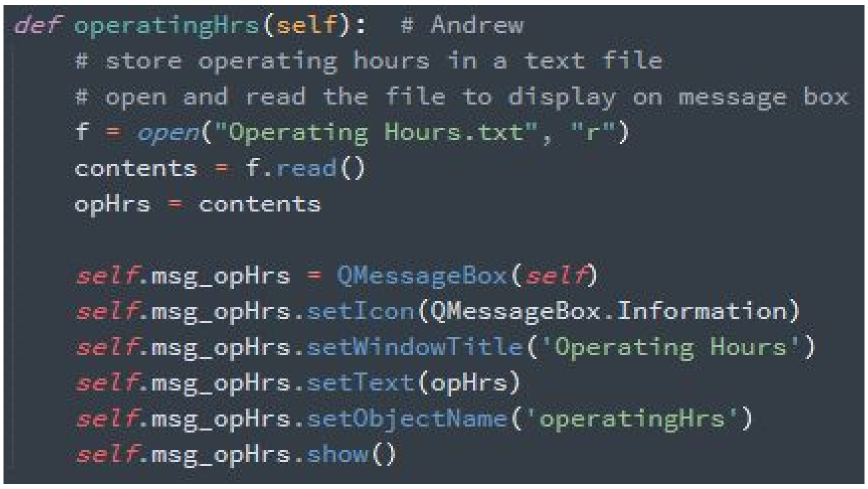
Code:



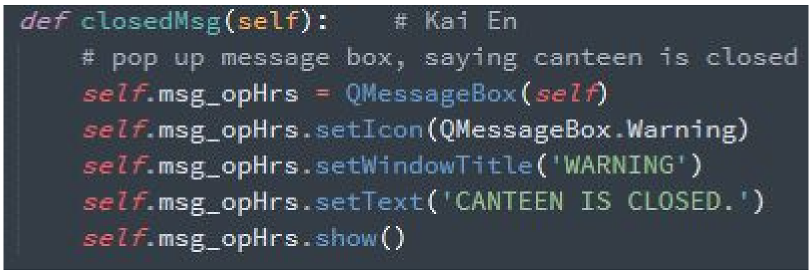
```python
def today_closed_or_open(self):   # Kai En
    # checks if the canteen is open today
    today = date.today()    # return today's local date
    day_value = today.weekday()  # return day of week as a value
    now = datetime.now()    # return current local date and time
    timeNow = now.time()    # create a time object for comparison
    cal = singapore.Singapore()  # get SG's Public Holidays

    # Mon is 0, Sunday is 6
    if day_value is 6:  # if its Sun, closed
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if not Sat & not a working day, closed
    elif day_value is not 5 and not cal.is_working_day(today):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if its Sat and time now is not btwn 7am-3pm
    elif day_value is 5 and not time(hour=7) <= timeNow <= time(hour=15):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if its Mon-Fri and time now is not btwn 7am-9pm
    elif 0 <= day_value <= 4 and not time(hour=7) <= timeNow <= time(hour=21):
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            self.closedMsg)
    # if code reaches here, canteen is open
    else:
        self.leftStack_stack0.btn_viewStores.clicked.connect(
            lambda: self.leftStack.setCurrentIndex(1))
```



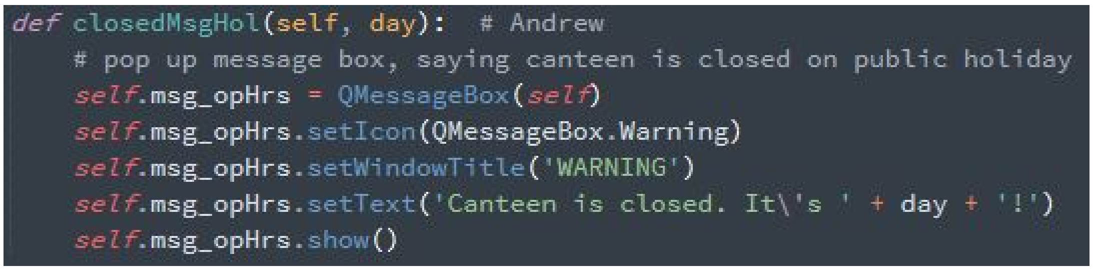
```python
def cfm_date(self):  # Andrew
    # check if store is closed
    # if selected date is a public holiday
    global logic
    selDate_datetime_date = self.cal.calendar.selectedDate().toPyDate() # retrieve date from CalendarUI
    selDate_val = selDate_datetime_date.weekday() # return day of week as int
    today = date.today()  # return today's local date
    selDate = self.cal.calendar.selectedDate() # date selected from calendar
    cbx = self.cal.combo_box.currentText()  # retrieve value from combo box
    cal = singapore.Singapore()  # get SG's Public Holidays

    # display selected date below listview
    self.rightStack.lbl_selDate.setText(selDate.toPyDate().strftime('%A, %d/%m/%Y') + " " + cbx)
    self.cal.hide() # hide calendar

    # checking for all holidays
    for var in cal.holidays():
        todaydate, day = var  # splitting value retrieved from holiday into todaydate and day
        logic = True
        if selDate == todaydate:
            self.closedMsgHol(day)
            logic = False
            break

    if selDate_val == 6:
        self.closedMsg()
    elif today == selDate_datetime_date:
        self.cal_today()
    elif logic == True:
        self.leftStack.setCurrentIndex(1)
        self.rightStack.textEdit.setText('`')
```



```python
def operatingHrs(self):  # Andrew
    # store operating hours in a text file
    # open and read the file to display on message box
    f = open("Operating Hours.txt", "r")
    contents = f.read()
    opHrs = contents

    self.msg_opHrs = QMessageBox(self)
    self.msg_opHrs.setIcon(QMessageBox.Information)
    self.msg_opHrs.setWindowTitle('Operating Hours')
    self.msg_opHrs.setText(opHrs)
    self.msg_opHrs.setObjectName('operatingHrs')
    self.msg_opHrs.show()
```



```python
def closedMsg(self):    # Kai En
    # pop up message box, saying canteen is closed
    self.msg_opHrs = QMessageBox(self)
    self.msg_opHrs.setIcon(QMessageBox.Warning)
    self.msg_opHrs.setWindowTitle('WARNING')
    self.msg_opHrs.setText('CANTEEN IS CLOSED.')
    self.msg_opHrs.show()
```



```
def closedMsgHol(self, day):  # Andrew
    # pop up message box, saying canteen is closed on public holiday
    self.msg_opHrs = QMessageBox(self)
    self.msg_opHrs.setIcon(QMessageBox.Warning)
    self.msg_opHrs.setWindowTitle('WARNING')
    self.msg_opHrs.setText('Canteen is closed. It\'s ' + day + '!')
    self.msg_opHrs.show()
```

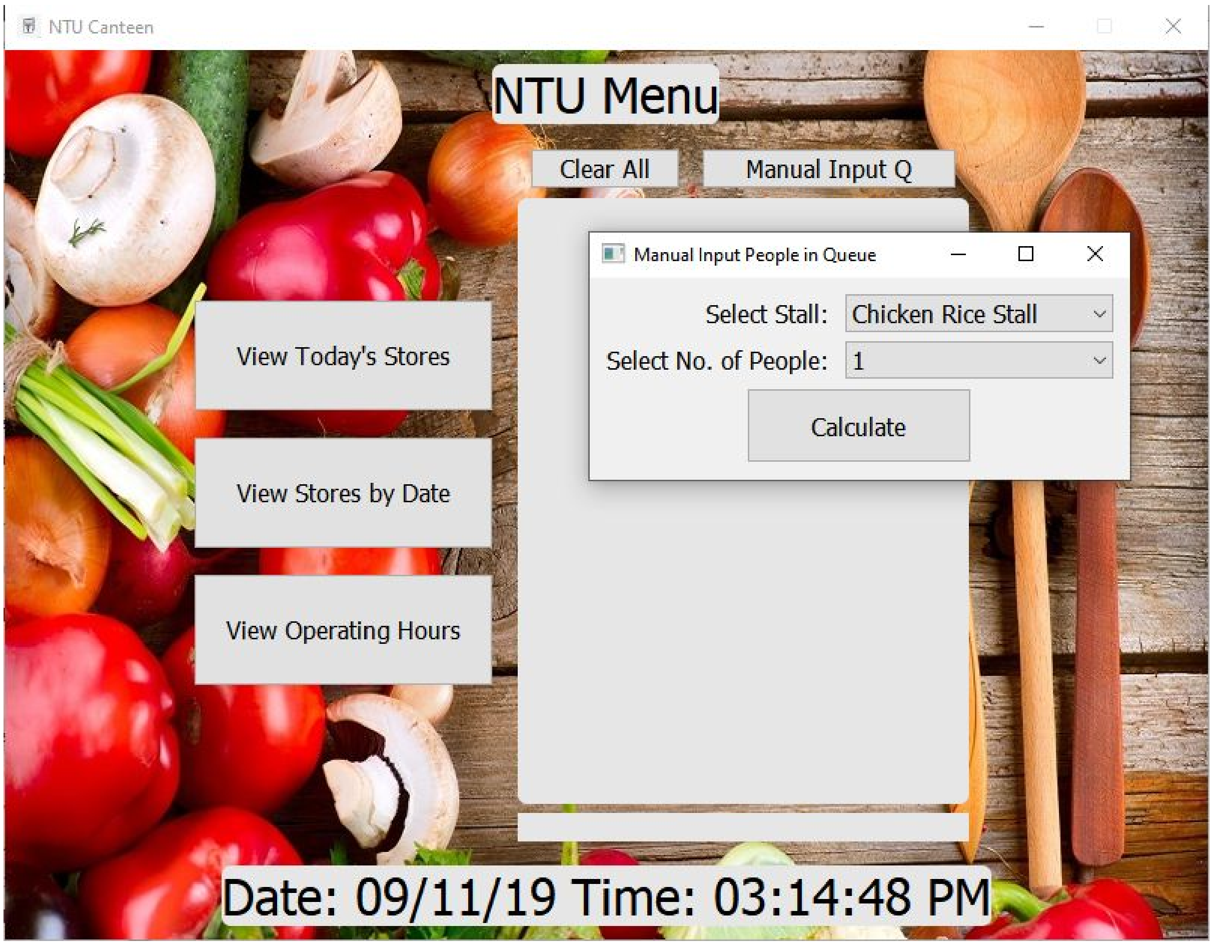
Description:

The first function checks if the canteen is open by verifying if it is a Sunday. If it is, the canteen is closed. Likewise, if it is a holiday, it remains closed. Lastly, if it is an operating day, but not within its operating hours, it stays closed too. After evaluating all possibilities of the canteen being closed, the else statement only executes if the previous conditions are not fulfilled which means the canteen is open. When it is closed, the interface will not allow the user to pick the stall menu to display.

operatingHrs() is the function used to connect the ‘View Operating Hours’ button for the user to see the operating hours of the canteen. This is done by opening and reading the contents of a text file.
closedMsg/closedMsgHol is the pop-up message used when the canteen is closed.

Program Testing

Case 1:



Select Stall: Chicken Rice Stall

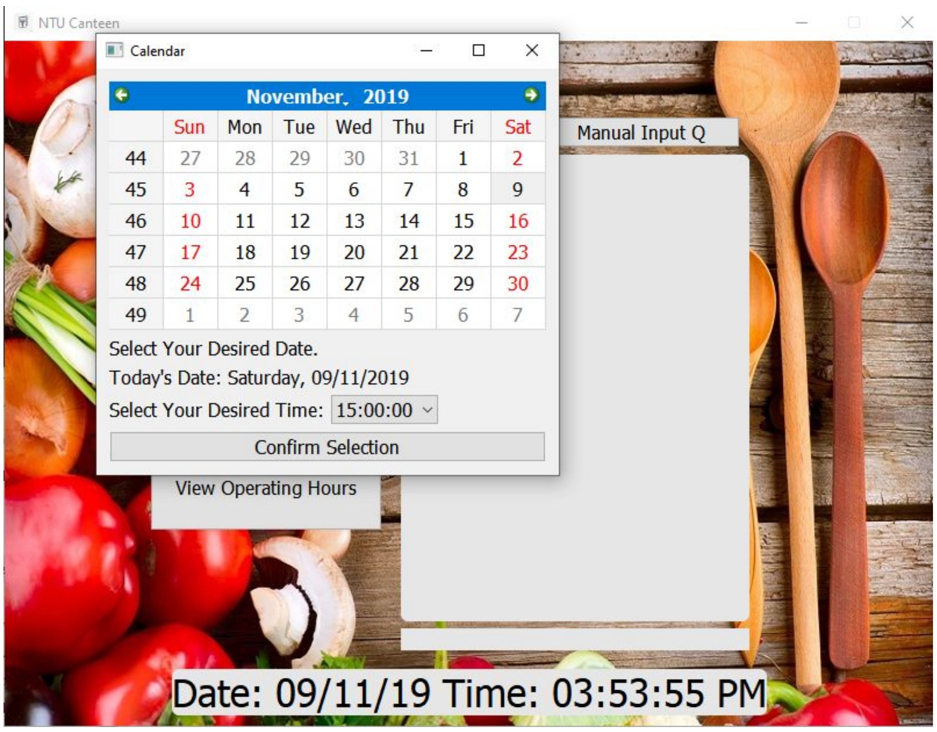
Select No. of People: 1

Date: 09/11/19 Time: 03:14:48 PM

Description:

Asking for users' manual inputs to calculate estimated waiting time may result in invalid inputs. Therefore, the program is restricted allowing users to select the number of people and the stall through a drop-down list. Though restricted, the number of people ranges to 100 to ensure wide enough of a range for users. This ensures the input will definitely be accepted.

Case 2:

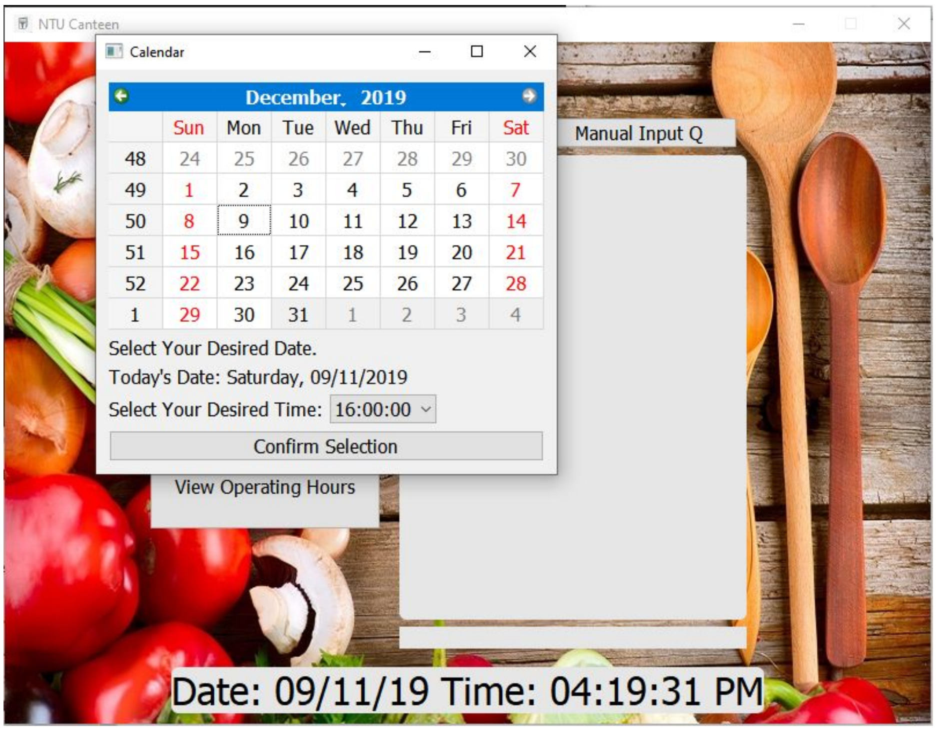


November, 2019

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
44	27	28	29	30	31	1	2
45	3	4	5	6	7	8	9
46	10	11	12	13	14	15	16
47	17	18	19	20	21	22	23
48	24	25	26	27	28	29	30
49	1	2	3	4	5	6	7

Select Your Desired Date.
Today's Date: Saturday, 09/11/2019
Select Your Desired Time: 15:00:00

Date: 09/11/19 Time: 03:53:55 PM



December, 2019

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
48	24	25	26	27	28	29	30
49	1	2	3	4	5	6	7
50	8	9	10	11	12	13	14
51	15	16	17	18	19	20	21
52	22	23	24	25	26	27	28
1	29	30	31	1	2	3	4

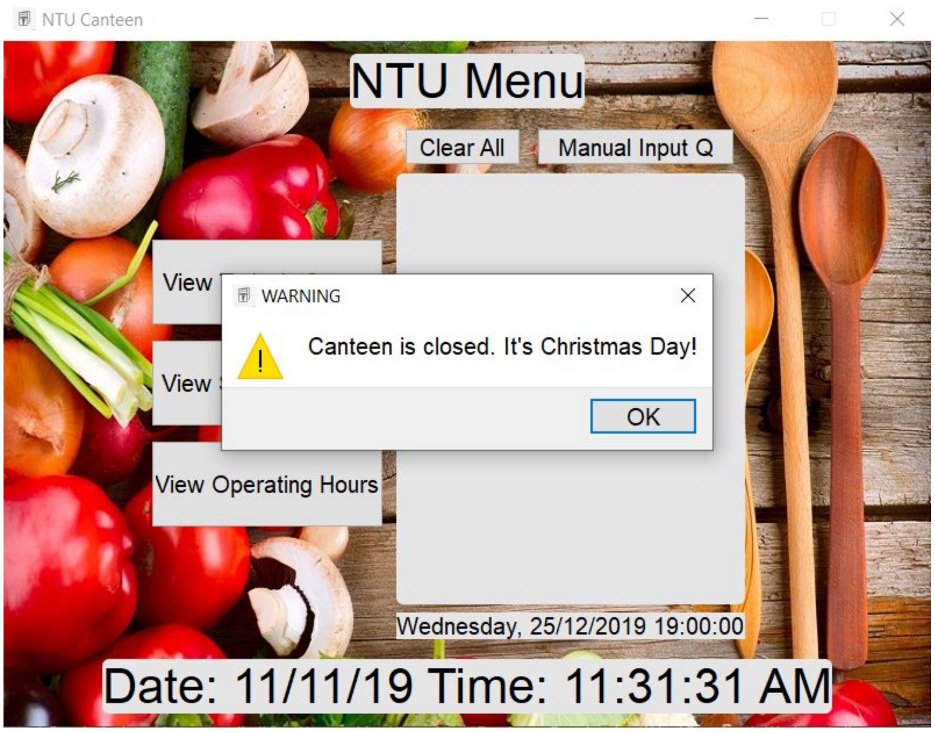
Select Your Desired Date.
Today's Date: Saturday, 09/11/2019
Select Your Desired Time: 16:00:00

Date: 09/11/19 Time: 04:19:31 PM

Description:

Similarly, for Feature D, the program is restricted to only allow the user to select the date via a calendar and the desired time through a drop-down list instead of manually typing it in.

Case 3:



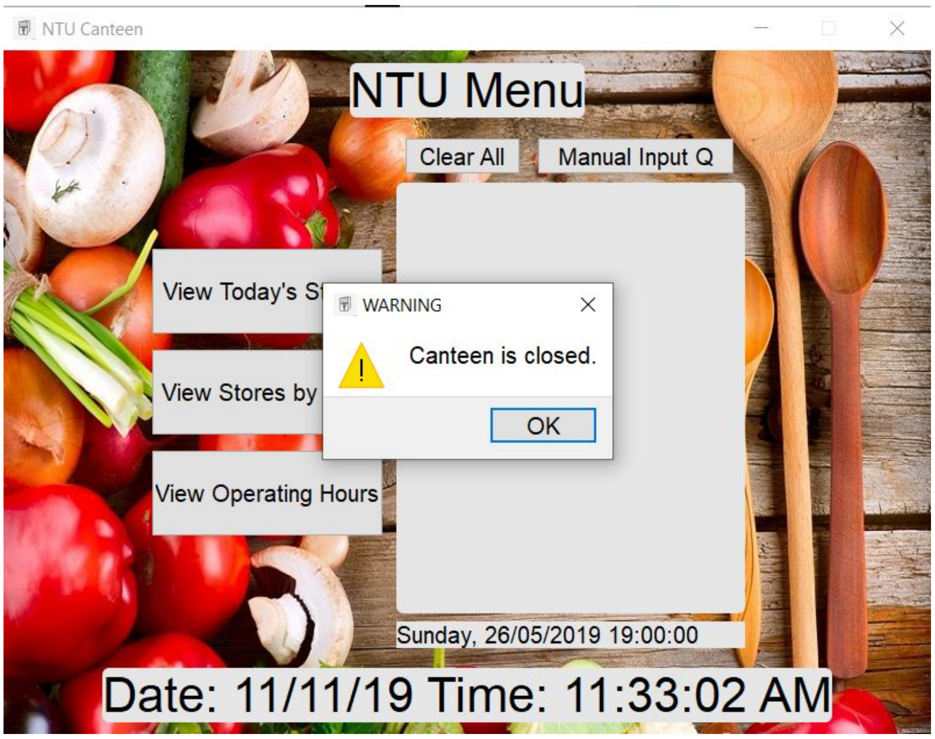
WARNING

Canteen is closed. It's Christmas Day!

OK

Wednesday, 25/12/2019 19:00:00

Date: 11/11/19 Time: 11:31:31 AM



WARNING

Canteen is closed.

OK

Sunday, 26/05/2019 19:00:00

Date: 11/11/19 Time: 11:33:02 AM

Description:

Validations are set where if users were to click on dates that are either Sunday or Public Holiday, they would be prompted a pop up message stating its closed.

Personal Reflections

Andrew:

Difficulties encountered is creation of validity of public holiday for the calendar and retrieving values from combo box.

Workalendar module stores a list of holiday formatted (datetime.date (2019, 1, 1), ‘New Year’). I needed only date from module to compare with the date selected by user to determine if it’s a public holiday. When retrieving values from module, the program would return both date and name. Using split function it didn’t work thus coming up with a method (todaydate, day = holiday). I am then able to compare the date from module and date selected by the user checking if it’s a public holiday. Another issue is the dropdown list in a format of ’08:00:00’. The value was needed for comparing to list different menuset. Python is unable to read in the format of ‘08:00:00’ by using replace method ‘:’ to ‘ ‘ and making the value into an integer to compare to state the menuset.

Improvement for this project would be looking towards implement clicking the text on the listview to display a pop up image of the item.

Kai En:

This project was a real eye-opener for me and using PyQt5 was a valuable experience. PyQt5 requires the use of object-oriented programming which I had to learn independently as it is not taught yet. Thus, this was a chance to better my understanding regarding the interaction of objects and classes in Python.

Additionally, I had a vague idea of PyQt5 when reading online tutorials but after countless trial-and-error sessions, I understood how to connect the different widgets, instantiate their classes to create them, and passing in self-defined functions to switch between & call the widgets for different interfaces. Herein lies the significance of compartmentalising code and importing them as modules to organise the program.

Another major problem I had occurred when I needed to retrieve values that I assigned in an instance. I realized that the values I assigned will be static as it is an attribute and thus will hold constant for an instance of the class. Hence, I had to change it to return a value as a function instead.

An improvement would be to get real-time data from the canteen regarding its queues instead of randomizing.

(1210 words)