SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CZ1003 Introduction to Computational Thinking

Mini Project

Real-time Canteen Information System

Lab Group: CM1

GROUP 1

Submitted By: Agnesh Ramesh

Anil Ankitha

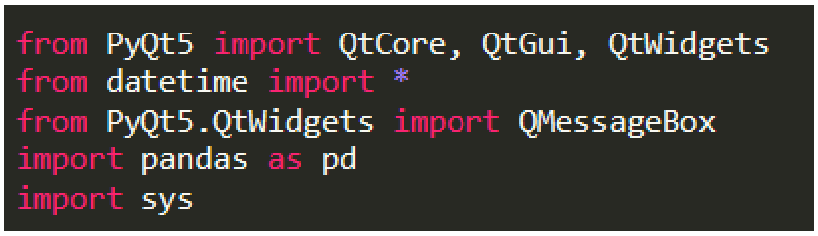
Chandrasekhar Aditya

Contents

INTRODUCTION

The application made is a Real-time Canteen Information System solely for North Spine Plaza in NTU. This application has a Graphic User Interface which has been made using the PyQt package in python. The menu of each stall is displayed in the interface based on the current date and time.
The program is user friendly and allows the user to customize the date and time, accordingly the menu of the stalls is displayed. Moreover, the user can calculate the waiting time by entering the number of people in front of the user in the queue. The application also allows the user to check the operating hours of each shop.

The GUI toolkits, database, external library or module that we used are:



```
from PyQt5 import QtCore, QtGui, QtWidgets
from datetime import *
from PyQt5.QtWidgets import QMessageBox
import pandas as pd
import sys
```

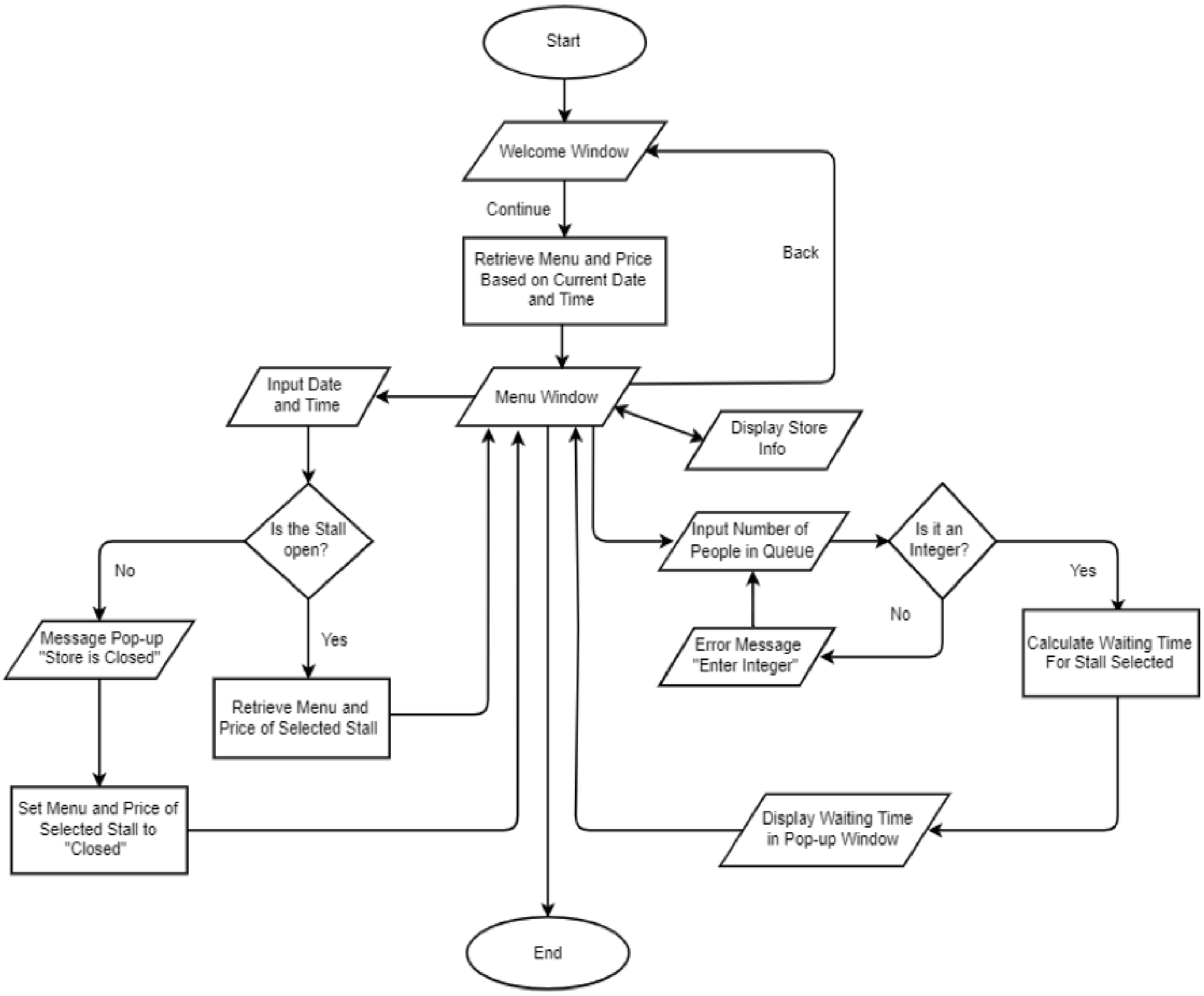
- PyQt5

  Used to design the main Graphic User Interface (GUI) for this application. This makes our GUI extremely user-friendly.
- Pandas

  Used to extract data from csv file in the form of a Data Frame. Information about stall menus was stored in this csv file depending on day and time for each stall. This data is converted into a list and then later into a dictionary.
- Datetime

  This module was used to retrieve the current date and time of the system, which was used to display stall menus accordingly. It is used to display a digital clock which updates displayed time constantly.

ALGORITHM DESIGN



Start
Welcome Window
Continue
Retrieve Menu and Price Based on Current Date and Time
Back
Menu Window
Input Date and Time
Display Store Info
Is the Stall open?
No
Yes
Message Pop-up "Store is Closed"
Retrieve Menu and Price of Selected Stall
Set Menu and Price of Selected Stall to "Closed"
Input Number of People in Queue
Is it an Integer?
Yes
No
Error Message "Enter Integer"
Calculate Waiting Time For Stall Selected
Display Waiting Time in Pop-up Window
End

Fig 1. Top Level flow chart

OVERVIEW OF THE PROGRAM

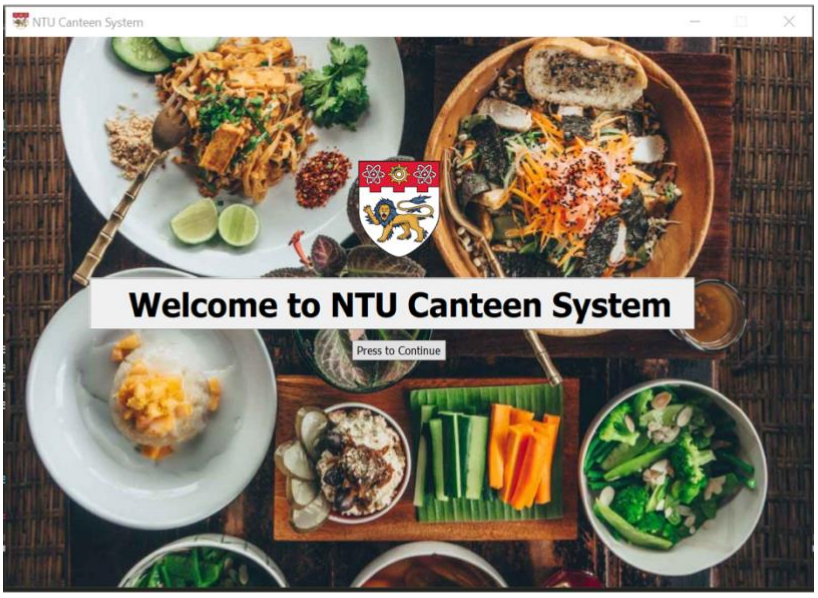
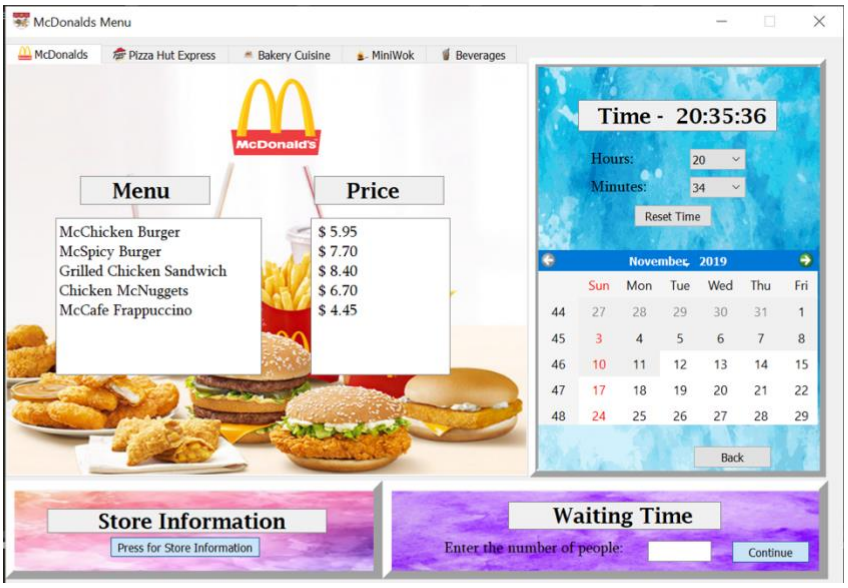


Fig 2. Main Window

This is the first window of the application. In this Graphical User Interface, there is one button which directs the user to the next window which has all the stall information.

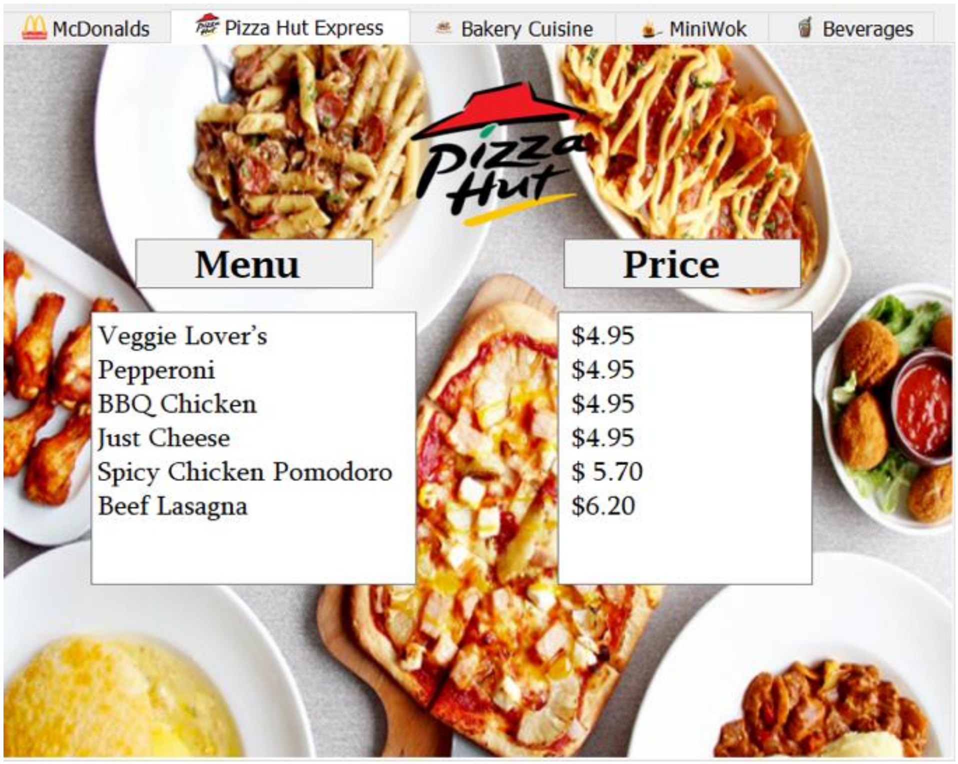


Menu	Price
McChicken Burger	$ 5.95
McSpicy Burger	$ 7.70
Grilled Chicken Sandwich	$ 8.40
Chicken McNuggets	$ 6.70
McCafe Frappuccino	$ 4.45

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

Fig 3. Main Menu

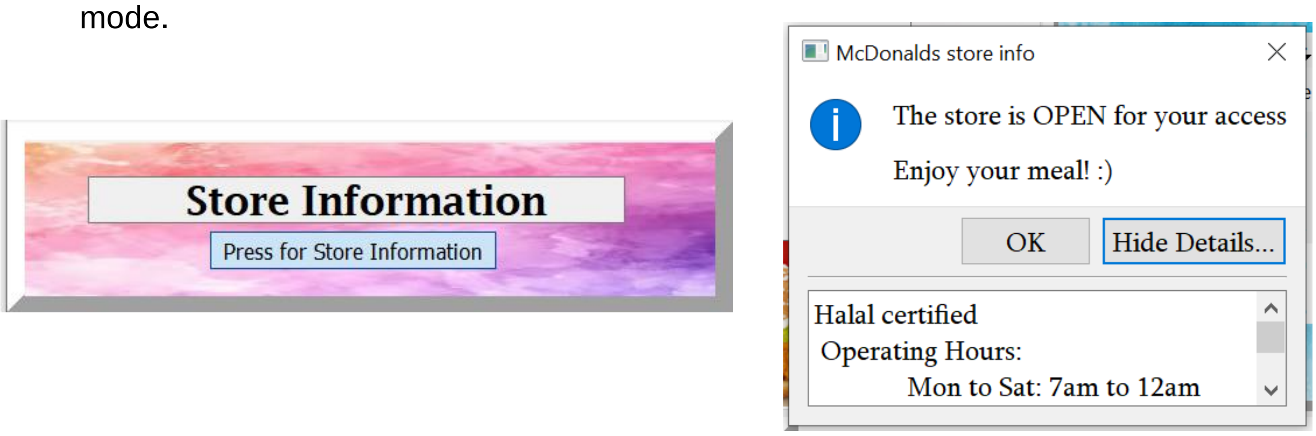
This is the second and the main window of the application. This window can be divided into 4 frames and the functionality of all these frames are integrated into a single window to make it more user friendly and easy to access.



Menu	Price
Veggie Lover's	$4.95
Pepperoni	$4.95
BBQ Chicken	$4.95
Just Cheese	$4.95
Spicy Chicken Pomodoro	$ 5.70
Beef Lasagna	$6.20

Fig 4. Menu frame

The first frame is used to shift between different stalls which can be accomplished by the Tab Widget. Each Tab Widget represents a stall and depending on the stall, the menu and the price of each item is displayed in a Text Edit widget in the read-only mode.



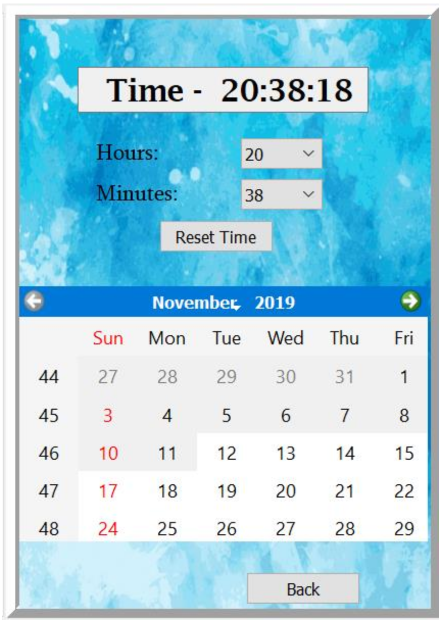
The store is OPEN for your access

Enjoy your meal! :)

Halal certified
Operating Hours:
Mon to Sat: 7am to 12am

Fig 5. Store Information frame

The second frame is used to display the stall information. It has a button in it and when clicked, a popup window shows up which indicates whether the shop is closed or open. the button in the popup window can be used to display store information such as stall timings and on which days of the week the stall is open.



Time - 20:38:18

Hours: 20

Minutes: 38

Reset Time

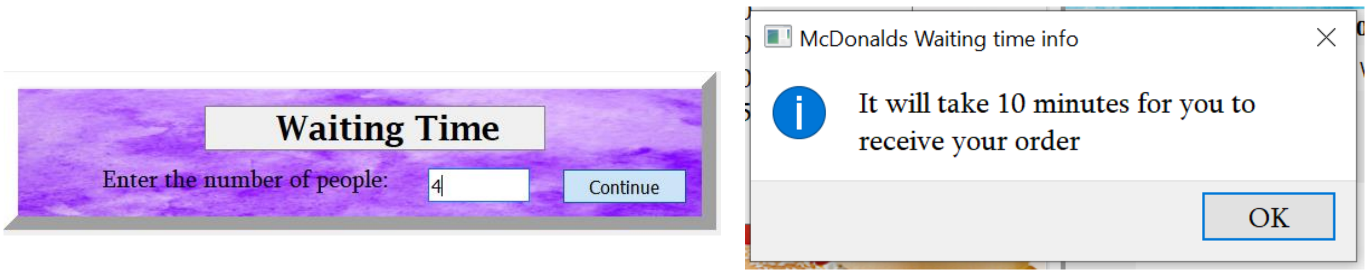
November 2019

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

Back

Fig 5. Date and Time frame

The third frame is used to check the menu for stalls at user defined and current date and time. The current time and date are automatically updated as shown. It consists of a calendar which allows user input to select the date. The frame contains 2 combo boxes which is used to input the hour and minutes. Depending on the values inputted in the combo boxes and calendar, the corresponding menu and prices are displayed.



Waiting Time

Enter the number of people: 4

Continue

McDonalds Waiting time info

It will take 10 minutes for you to receive your order

OK

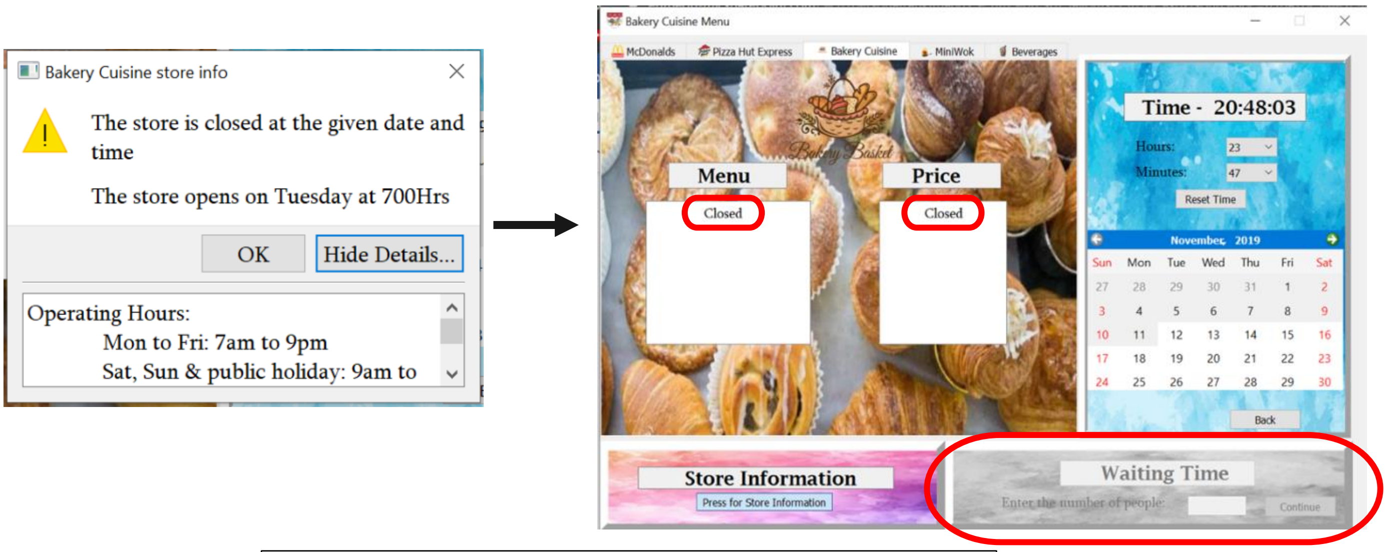
Fig 6. Waiting Time frame

The fourth frame is used by the user to input the number of people standing ahead of the user in the queue and depending on the integer given, the estimated waiting time is displayed in a popup window.

ADDITIONAL FEATURES

The additional features in our program for handling errors encountered and making the program more efficient are:

1. The date and time that has already elapsed cannot be accessed.



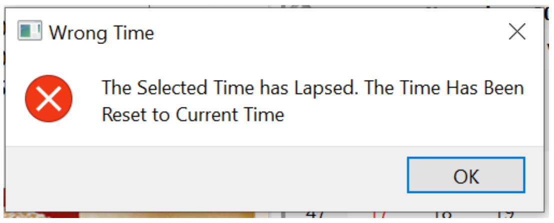
Bakery Cuisine store info

The store is closed at the given date and time

The store opens on Tuesday at 700Hrs

Operating Hours:
Mon to Fri: 7am to 9pm
Sat, Sun & public holiday: 9am to

Fig 7. User defined date and time

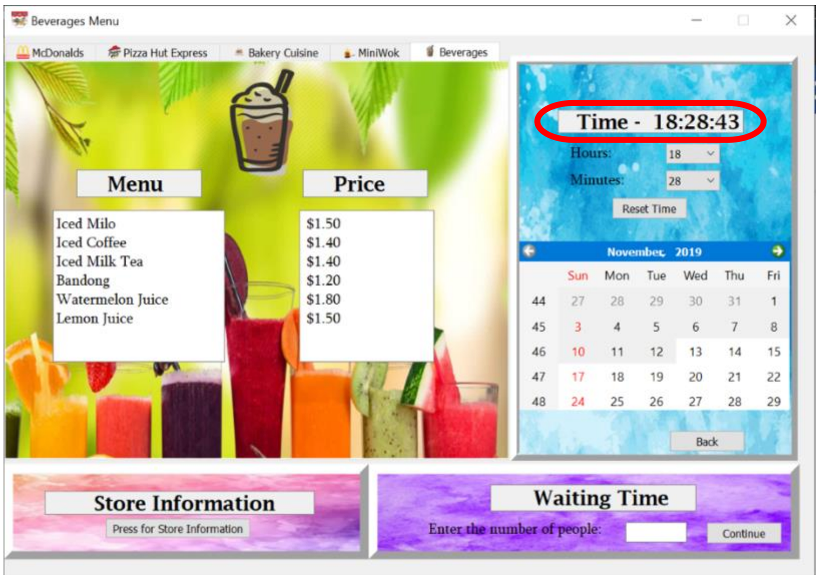


Wrong Time

The Selected Time has Lapsed. The Time Has Been Reset to Current Time

Fig 8. User defined time of the current day

2. The digital clock which constantly updates its time

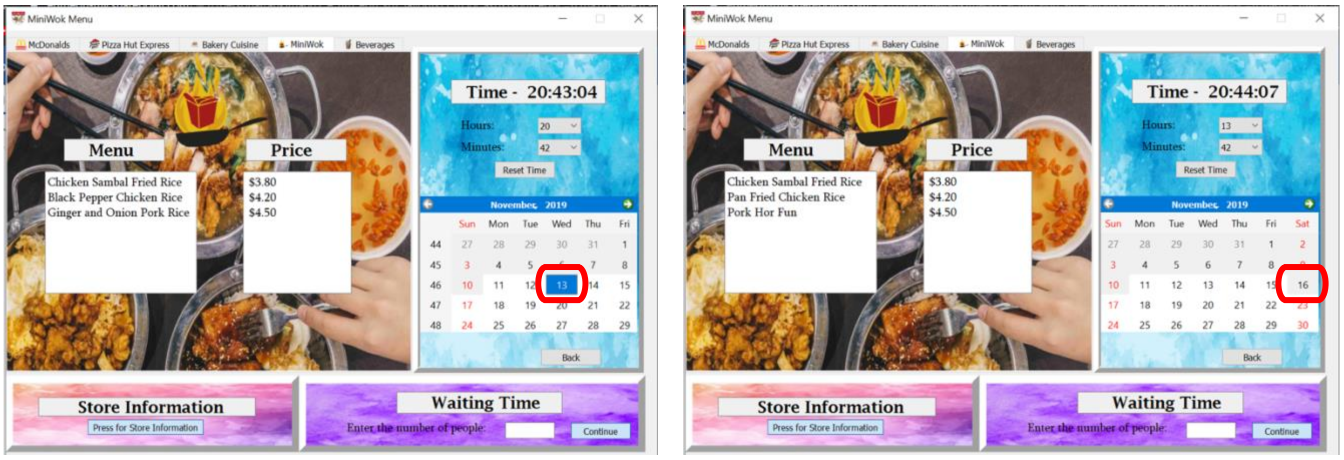


Beverages Menu

Menu	Price
Iced Milo	$1.50
Iced Coffee	$1.40
Iced Milk Tea	$1.40
Bandong	$1.20
Watermelon Juice	$1.80
Lemon Juice	$1.50

Fig 9. Updating clock

3. Different menus for different times of the day



Time - 20:43:04

Hours: 20
Minutes: 42

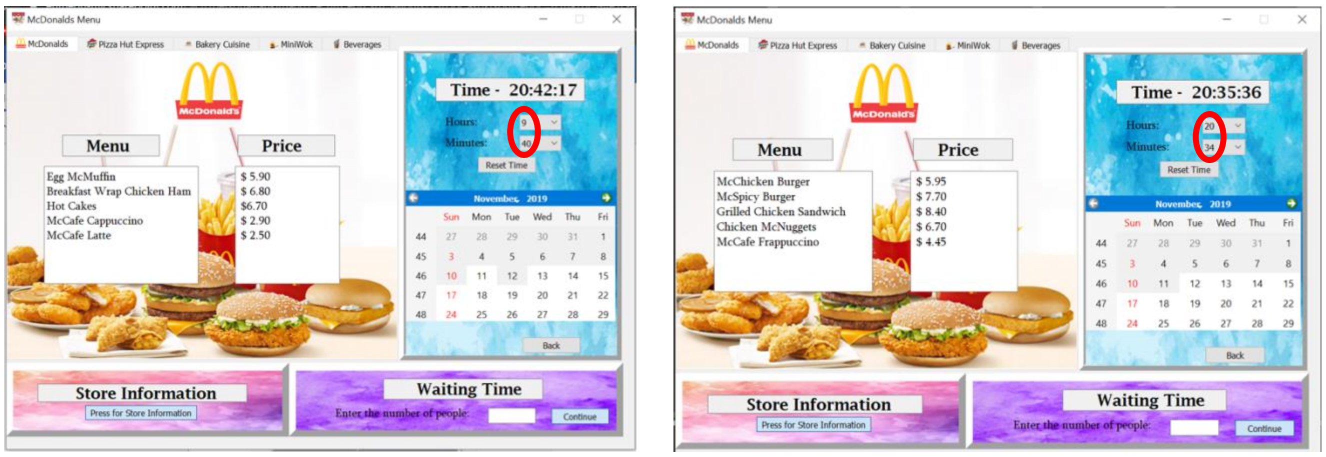
Menu	Price
Chicken Sambal Fried Rice	$3.80
Black Pepper Chicken Rice	$4.20
Ginger and Onion Pork Rice	$4.50

Time - 20:44:07

Hours: 13
Minutes: 42

Menu	Price
Chicken Sambal Fried Rice	$3.80
Pan Fried Chicken Rice	$4.20
Pork Hor Fun	$4.50

Fig 10. Weekday and Weekend Miniwok Menu



Time - 20:42:17

Hours: 9
Minutes: 40

Menu	Price
Egg McMuffin	$ 5.90
Breakfast Wrap Chicken Ham	$ 6.80
Hot Cakes	$6.70
McCafe Cappuccino	$ 2.90
McCafe Latte	$ 2.50

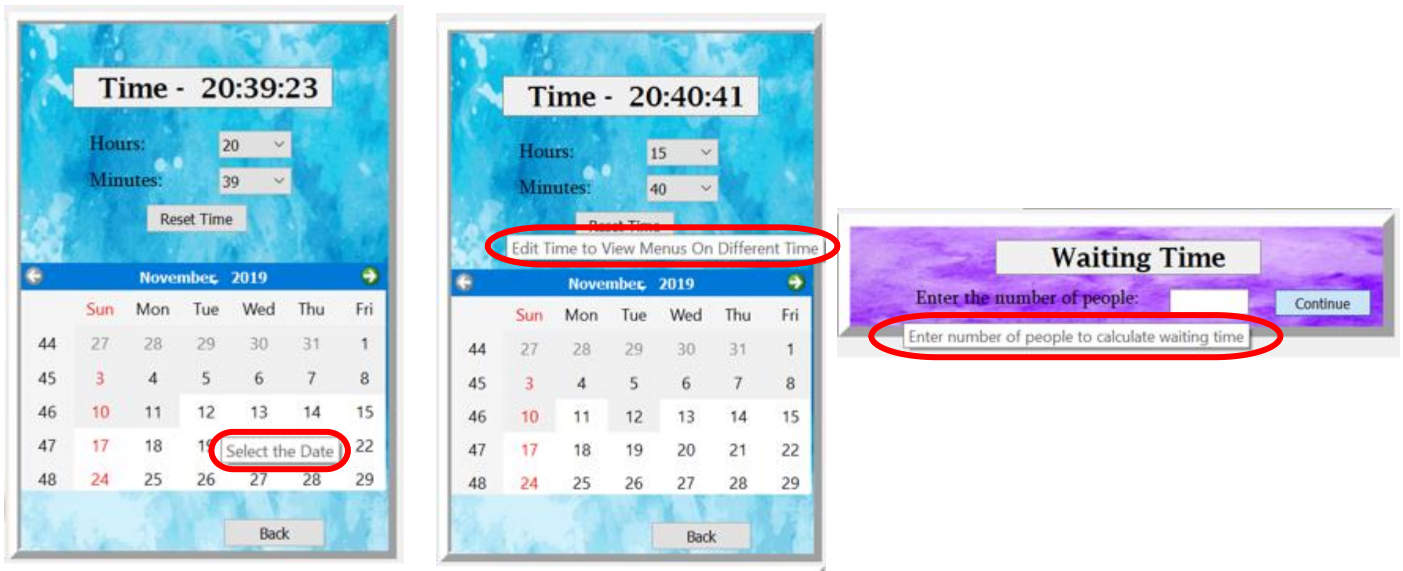
Time - 20:35:36

Hours: 20
Minutes: 34

Menu	Price
McChicken Burger	$ 5.95
McSpicy Burger	$ 7.70
Grilled Chicken Sandwich	$ 8.40
Chicken McNuggets	$ 6.70
McCafe Frappuccino	$ 4.45

Fig 11. Morning and Afternoon McDonalds Menu

4. Tool tip is used to guide the user



Time - 20:39:23

Hours: 20
Minutes: 39

Select the Date

Time - 20:40:41

Hours: 15
Minutes: 40

Edit Time to View Menus On Different Time

Enter the number of people:

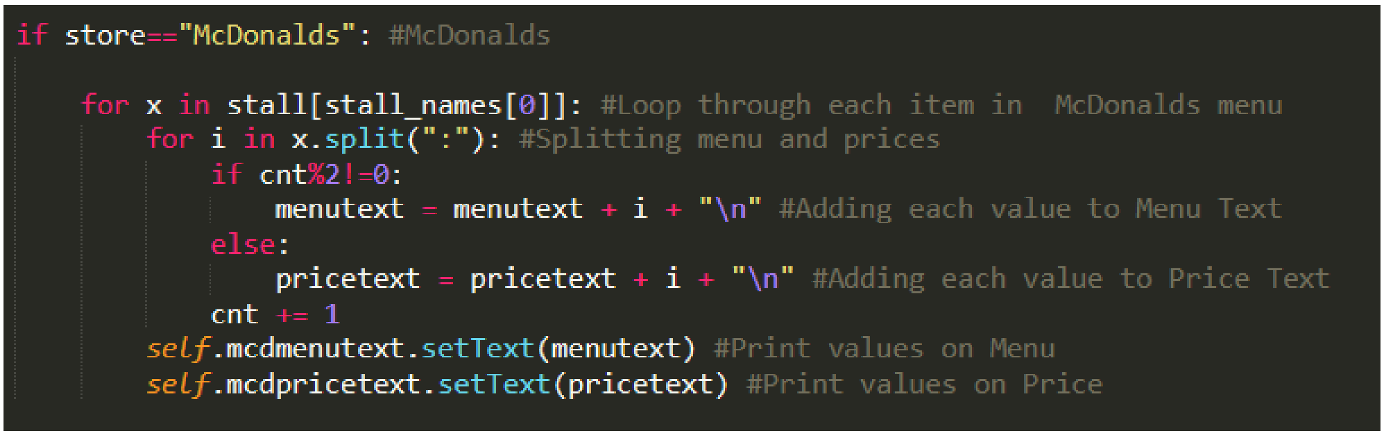
Enter number of people to calculate waiting time

Fig 12. Tool tips

USAGE OF DATA STRUCTURES

The data structure used in the program include:

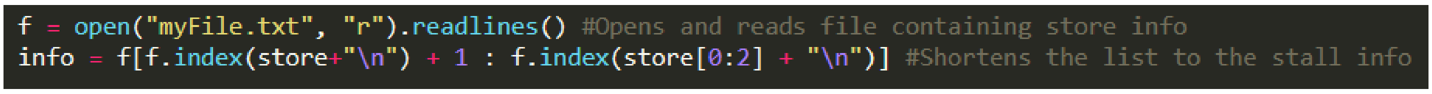
1. Strings: Number of string functions and methods like split(), indexing, concatenation where used.



```python
if store=="McDonalds": #McDonalds

    for x in stall[stall_names[0]]: #Loop through each item in  McDonalds menu
        for i in x.split(":"): #Splitting menu and prices
            if cnt%2!=0:
                menutext = menutext + i + "\n" #Adding each value to Menu Text
            else:
                pricetext = pricetext + i + "\n" #Adding each value to Price Text
            cnt += 1
        self.mcdmenutext.setText(menutext) #Print values on Menu
        self.mcdpricetext.setText(pricetext) #Print values on Price
```

Fig 13. Example of string function .split(), string concatenation

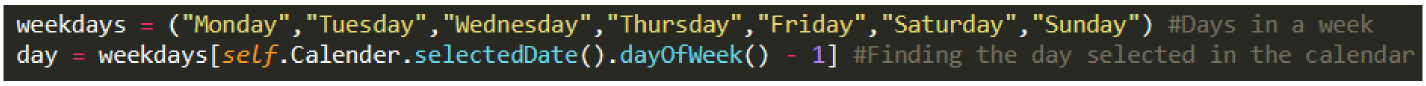
2. Lists: Number of list functions like slicing, .join(), .pop() and remove functions.



```python
f = open("myFile.txt", "r").readlines() #Opens and reads file containing store info
info = f[f.index(store+"\n") + 1 : f.index(store[0:2] + "\n")] #Shortens the list to the stall info
```

Fig 14. Example of List

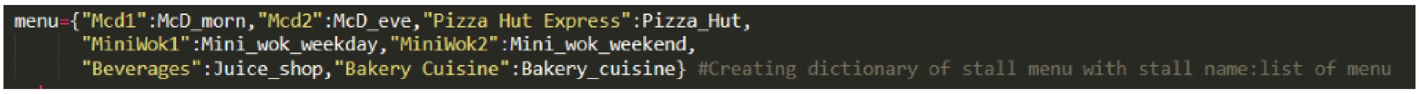
3. Tuples: Used indexing for retrieving weekdays and manipulating menu and prices data accordingly.



```python
weekdays = ("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday") #Days in a week
day = weekdays[self.Calender.selectedDate().dayOfWeek() - 1] #Finding the day selected in the calendar
```

Fig 15. Example of Tuple

4. Dictionaries: Used to store information such as operating hours, menu and these dictionaries were used to retrieve according to date and time.



```python
menu={"Mcd1":McD_morn,"Mcd2":McD_eve,"Pizza Hut Express":Pizza_Hut,
      "MiniWok1":Mini_wok_weekday,"MiniWok2":Mini_wok_weekend,
      "Beverages":Juice_shop,"Bakery Cuisine":Bakery_cuisine} #Creating dictionary of stall menu with stall name:list of menu
```

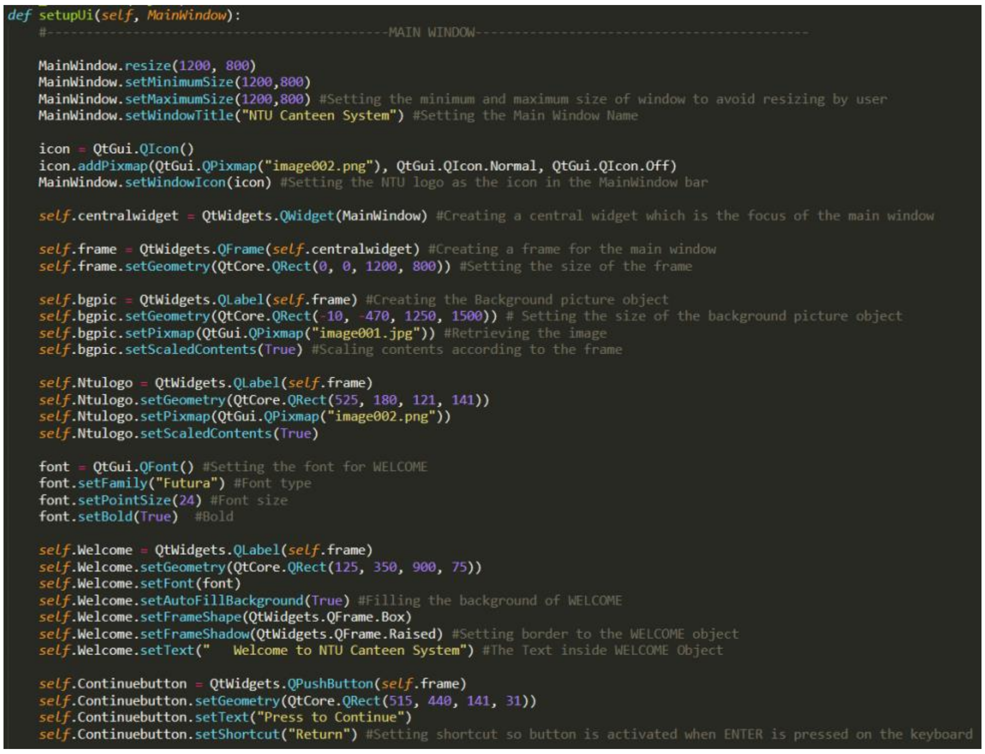
Fig 16. Example of Dictionary

We have also used a combination of data structures like a list inside a dictionary to store information.

USER DEFINED FUNCTIONS

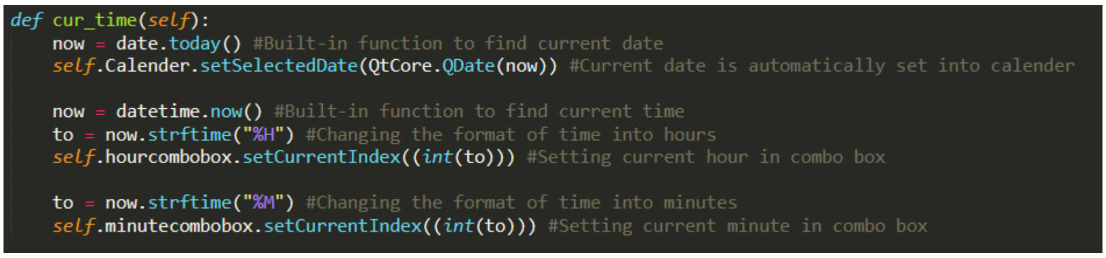
1. **setupUi():** Used to setup the GUI (front end) of the mini project.

In this function, the Welcome window and Main menu window are being set up along with other contents and specifications such as text, font, alignment, images and widgets. This function defines how the GUI looks.



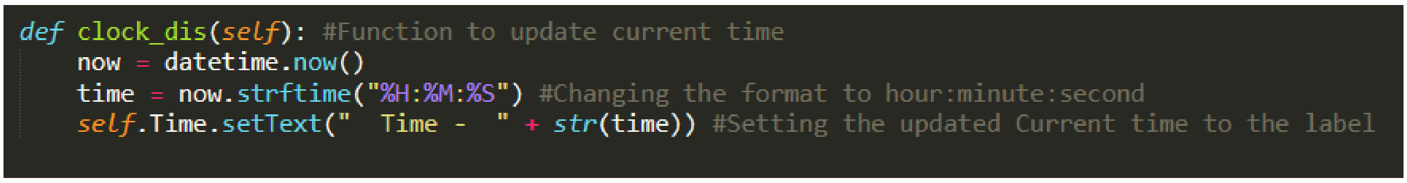
```python
def setupUi(self, MainWindow):
    #------------------------------------------MAIN WINDOW------------------------------------------

    MainWindow.resize(1200, 800)
    MainWindow.setMinimumSize(1200,800)
    MainWindow.setMaximumSize(1200,800) #Setting the minimum and maximum size of window to avoid resizing by user
    MainWindow.setWindowTitle("NTU Canteen System") #Setting the Main Window Name

    icon = QtGui.QIcon()
    icon.addPixmap(QtGui.QPixmap("image002.png"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
    MainWindow.setWindowIcon(icon) #Setting the NTU logo as the icon in the MainWindow bar

    self.centralwidget = QtWidgets.QWidget(MainWindow) #Creating a central widget which is the focus of the main window

    self.frame = QtWidgets.QFrame(self.centralwidget) #Creating a frame for the main window
    self.frame.setGeometry(QtCore.QRect(0, 0, 1200, 800)) #Setting the size of the frame

    self.bgpic = QtWidgets.QLabel(self.frame) #Creating the Background picture object
    self.bgpic.setGeometry(QtCore.QRect(-10, -470, 1250, 1500)) # Setting the size of the background picture object
    self.bgpic.setPixmap(QtGui.QPixmap("image001.jpg")) #Retrieving the image
    self.bgpic.setScaledContents(True) #Scaling contents according to the frame

    self.Ntulogo = QtWidgets.QLabel(self.frame)
    self.Ntulogo.setGeometry(QtCore.QRect(525, 180, 121, 141))
    self.Ntulogo.setPixmap(QtGui.QPixmap("image002.png"))
    self.Ntulogo.setScaledContents(True)

    font = QtGui.QFont() #Setting the font for WELCOME
    font.setFamily("Futura") #Font type
    font.setPointSize(24) #Font size
    font.setBold(True)  #Bold

    self.Welcome = QtWidgets.QLabel(self.frame)
    self.Welcome.setGeometry(QtCore.QRect(125, 350, 900, 75))
    self.Welcome.setFont(font)
    self.Welcome.setAutoFillBackground(True) #Filling the background of WELCOME
    self.Welcome.setFrameShape(QtWidgets.QFrame.Box)
    self.Welcome.setFrameShadow(QtWidgets.QFrame.Raised) #Setting border to the WELCOME object
    self.Welcome.setText("   Welcome to NTU Canteen System") #The Text inside WELCOME Object

    self.Continuebutton = QtWidgets.QPushButton(self.frame)
    self.Continuebutton.setGeometry(QtCore.QRect(515, 440, 141, 31))
    self.Continuebutton.setText("Press to Continue")
    self.Continuebutton.setShortcut("Return") #Setting shortcut so button is activated when ENTER is pressed on the keyboard
```

2. **cur_time():** Function to set current date and time to the calendar and hour and minute combo boxes. This function retrieves current date and time from the system using datetime library of python.



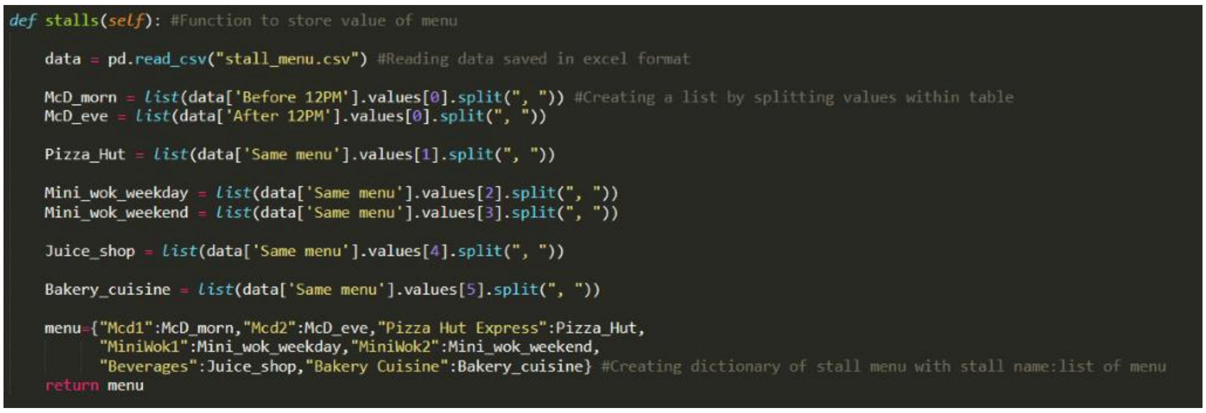
```python
def cur_time(self):
    now = date.today() #Built-in function to find current date
    self.Calender.setSelectedDate(QtCore.QDate(now)) #Current date is automatically set into calender

    now = datetime.now() #Built-in function to find current time
    to = now.strftime("%H") #Changing the format of time into hours
    self.hourcombobox.setCurrentIndex((int(to))) #Setting current hour in combo box

    to = now.strftime("%M") #Changing the format of time into minutes
    self.minutecombobox.setCurrentIndex((int(to))) #Setting current minute in combo box
```

3. **clock_dis():** This function is called for every time out of a Qtimer widgets. This function updates the current system time to a label using the datetime module of python.



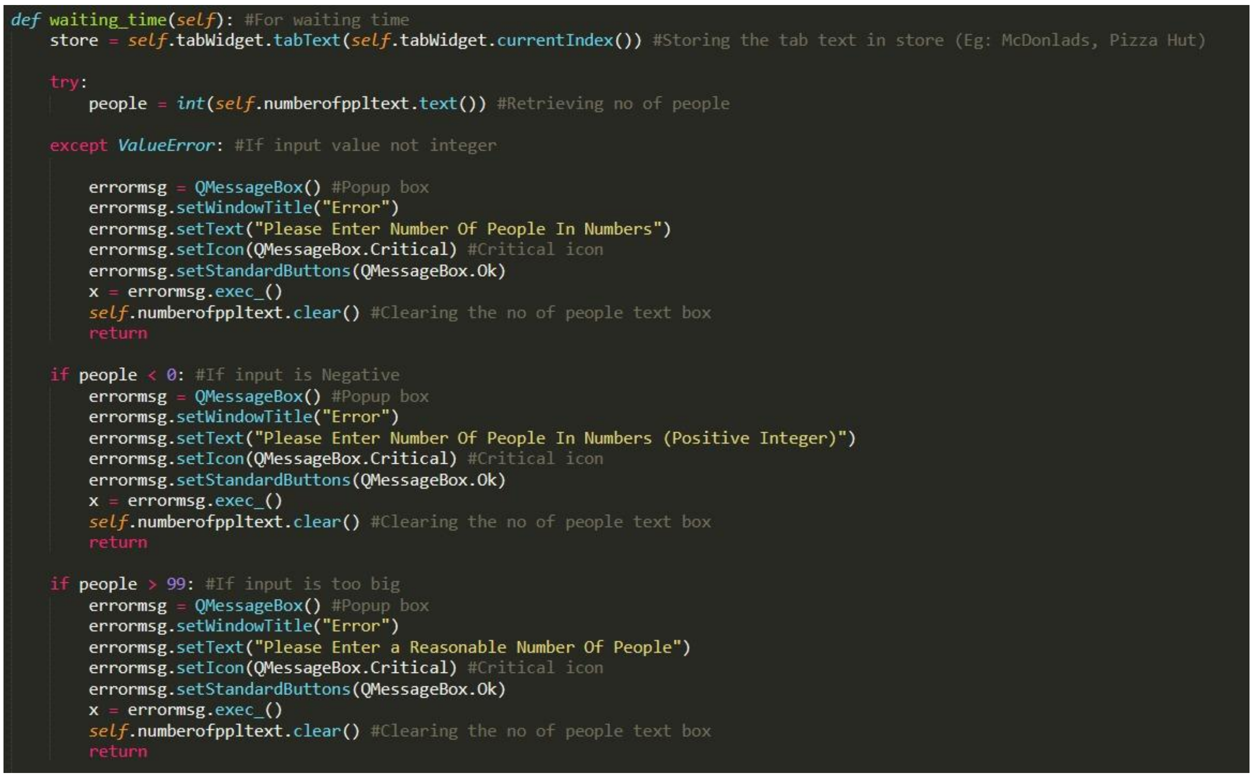
```python
def clock_dis(self): #Function to update current time
    now = datetime.now()
    time = now.strftime("%H:%M:%S") #Changing the format to hour:minute:second
    self.Time.setText("  Time -  " + str(time)) #Setting the updated Current time to the label
```

4. **stalls():** Function to retrieve stall menus from the file. Stall menu and prices are retrieved from the stall_menu.csv file using this function with the help of pandas and returned in the form of a dictionary.

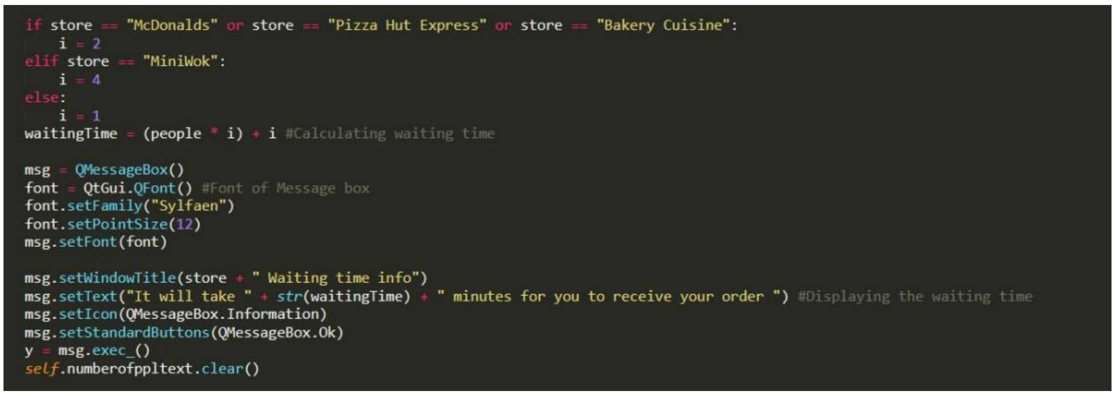


```python
def stalls(self): #Function to store value of menu

    data = pd.read_csv("stall_menu.csv") #Reading data saved in excel format

    McD_morn = list(data['Before 12PM'].values[0].split(", ")) #Creating a list by splitting values within table
    McD_eve = list(data['After 12PM'].values[0].split(", "))

    Pizza_Hut = list(data['Same menu'].values[1].split(", "))

    Mini_wok_weekday = list(data['Same menu'].values[2].split(", "))
    Mini_wok_weekend = list(data['Same menu'].values[3].split(", "))

    Juice_shop = list(data['Same menu'].values[4].split(", "))

    Bakery_cuisine = list(data['Same menu'].values[5].split(", "))

    menu={"Mcd1":McD_morn,"Mcd2":McD_eve,"Pizza Hut Express":Pizza_Hut,
          "MiniWok1":Mini_wok_weekday,"MiniWok2":Mini_wok_weekend,
          "Beverages":Juice_shop,"Bakery Cuisine":Bakery_cuisine} #Creating dictionary of stall menu with stall name:list of menu
    return menu
```

5. **waiting_time():** Function to calculate waiting time. This function contains a try and except block to verify whether the input is a positive integer.

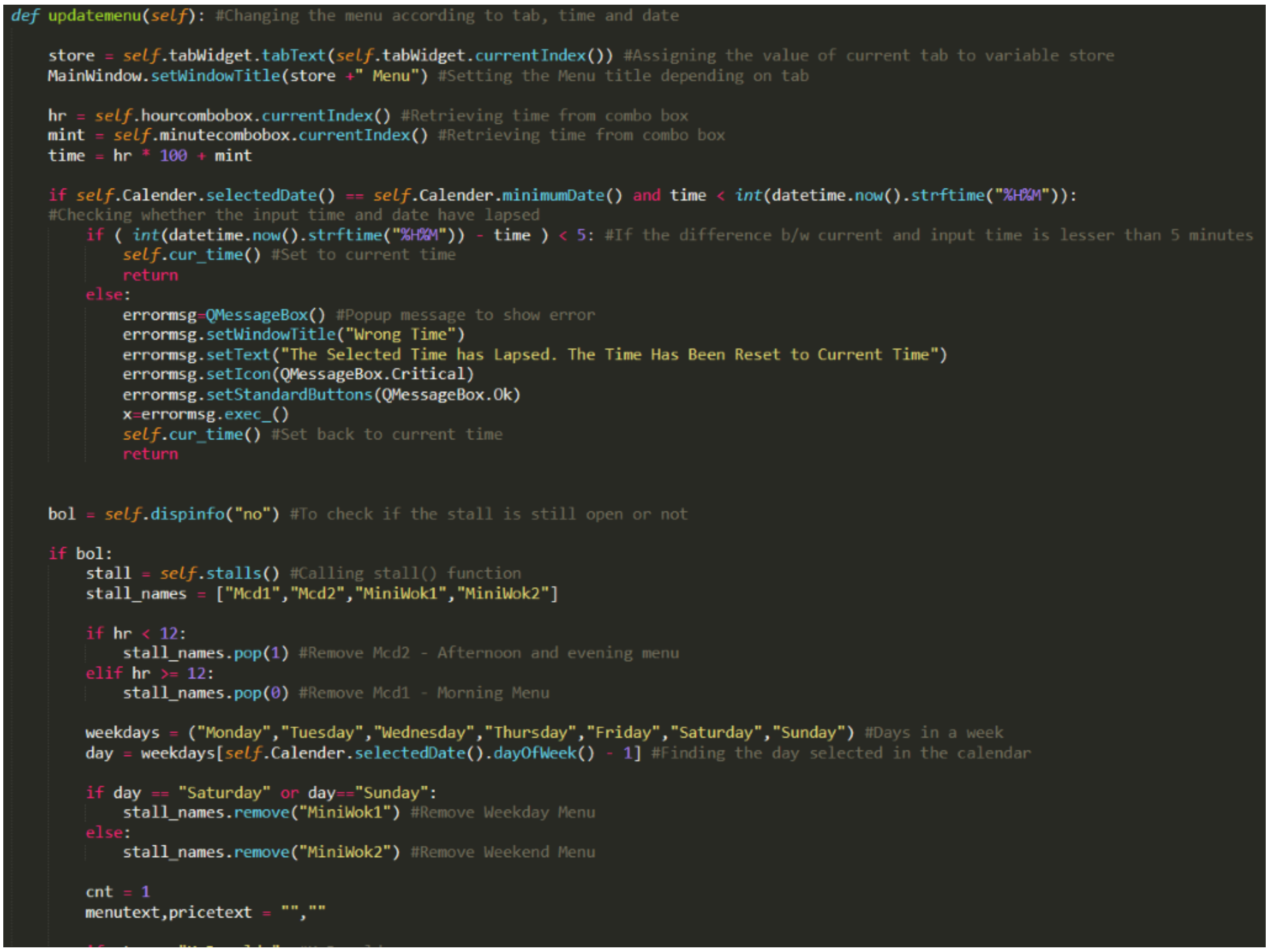


```python
def waiting_time(self): #For waiting time
    store = self.tabWidget.tabText(self.tabWidget.currentIndex()) #Storing the tab text in store (Eg: McDonlads, Pizza Hut)

    try:
        people = int(self.numberofppltext.text()) #Retrieving no of people

    except ValueError: #If input value not integer

        errormsg = QMessageBox() #Popup box
        errormsg.setWindowTitle("Error")
        errormsg.setText("Please Enter Number Of People In Numbers")
        errormsg.setIcon(QMessageBox.Critical) #Critical icon
        errormsg.setStandardButtons(QMessageBox.Ok)
        x = errormsg.exec_()
        self.numberofppltext.clear() #Clearing the no of people text box
        return

    if people < 0: #If input is Negative
        errormsg = QMessageBox() #Popup box
        errormsg.setWindowTitle("Error")
        errormsg.setText("Please Enter Number Of People In Numbers (Positive Integer)")
        errormsg.setIcon(QMessageBox.Critical) #Critical icon
        errormsg.setStandardButtons(QMessageBox.Ok)
        x = errormsg.exec_()
        self.numberofppltext.clear() #Clearing the no of people text box
        return

    if people > 99: #If input is too big
        errormsg = QMessageBox() #Popup box
        errormsg.setWindowTitle("Error")
        errormsg.setText("Please Enter a Reasonable Number Of People")
        errormsg.setIcon(QMessageBox.Critical) #Critical icon
        errormsg.setStandardButtons(QMessageBox.Ok)
        x = errormsg.exec_()
        self.numberofppltext.clear() #Clearing the no of people text box
        return
```



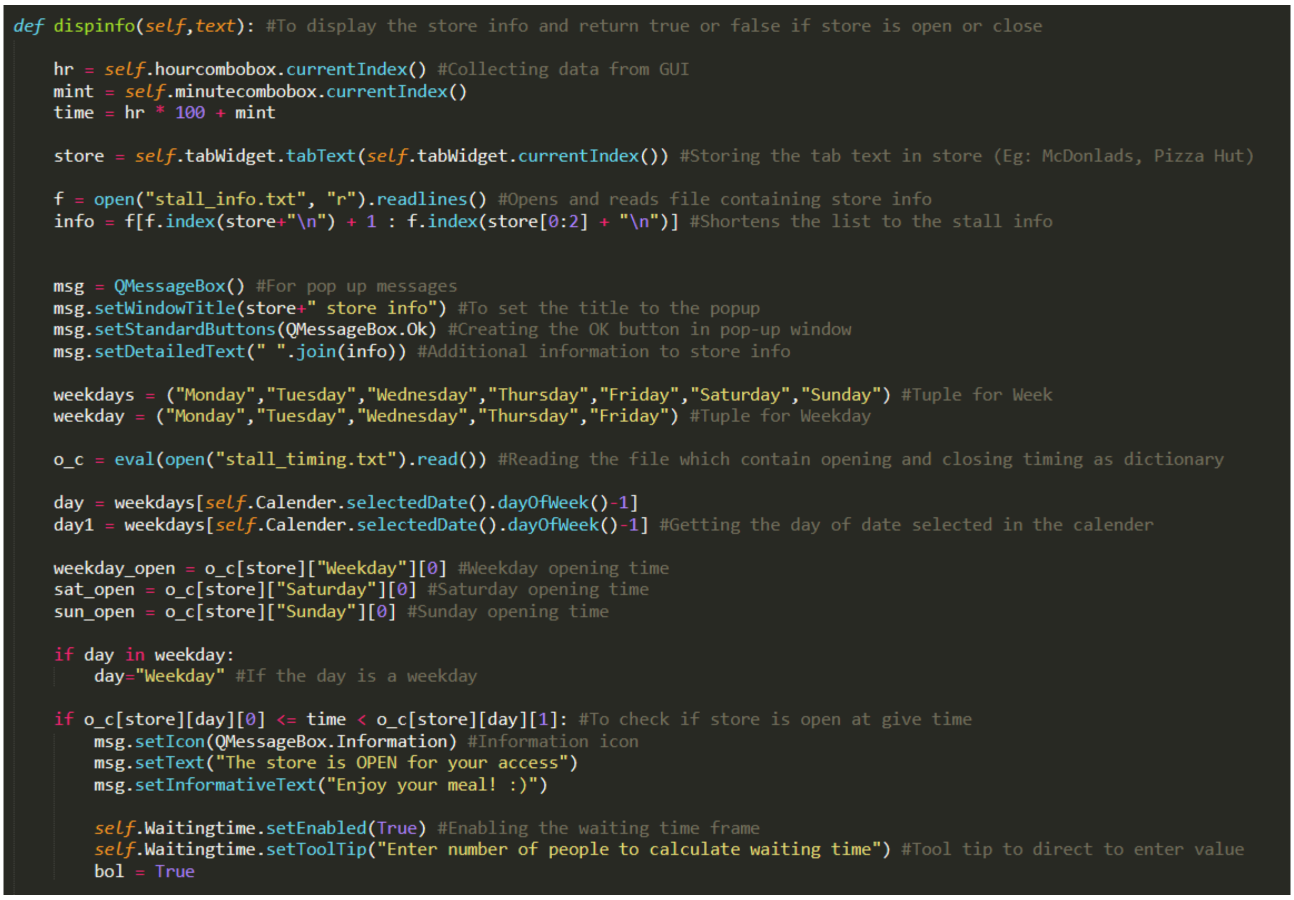
```python
if store == "McDonalds" or store == "Pizza Hut Express" or store == "Bakery Cuisine":
    i = 2
elif store == "MiniWok":
    i = 4
else:
    i = 1
waitingTime = (people * i) + i #Calculating waiting time

msg = QMessageBox()
font = QtGui.QFont() #Font of Message box
font.setFamily("Sylfaen")
font.setPointSize(12)
msg.setFont(font)

msg.setWindowTitle(store + " Waiting time info")
msg.setText("It will take " + str(waitingTime) + " minutes for you to receive your order ") #Displaying the waiting time
msg.setIcon(QMessageBox.Information)
msg.setStandardButtons(QMessageBox.Ok)
y = msg.exec_()
self.numberofppltext.clear()
```

6. **updatemenu():** Function to update items and prices according to the stall selected and given date and time. This function calls the disp_info function to check if the stall is open or not and displays the menu and price from the stall function accordingly.



```python
def updatemenu(self): #Changing the menu according to tab, time and date

    store = self.tabWidget.tabText(self.tabWidget.currentIndex()) #Assigning the value of current tab to variable store
    MainWindow.setWindowTitle(store +" Menu") #Setting the Menu title depending on tab

    hr = self.hourcombobox.currentIndex() #Retrieving time from combo box
    mint = self.minutecombobox.currentIndex() #Retrieving time from combo box
    time = hr * 100 + mint

    if self.Calender.selectedDate() == self.Calender.minimumDate() and time < int(datetime.now().strftime("%H%M")):
    #Checking whether the input time and date have lapsed
        if ( int(datetime.now().strftime("%H%M")) - time ) < 5: #If the difference b/w current and input time is lesser than 5 minutes
            self.cur_time() #Set to current time
            return
        else:
            errormsg=QMessageBox() #Popup message to show error
            errormsg.setWindowTitle("Wrong Time")
            errormsg.setText("The Selected Time has Lapsed. The Time Has Been Reset to Current Time")
            errormsg.setIcon(QMessageBox.Critical)
            errormsg.setStandardButtons(QMessageBox.Ok)
            x=errormsg.exec_()
            self.cur_time() #Set back to current time
            return


    bol = self.dispinfo("no") #To check if the stall is still open or not

    if bol:
        stall = self.stalls() #Calling stall() function
        stall_names = ["Mcd1","Mcd2","MiniWok1","MiniWok2"]

        if hr < 12:
            stall_names.pop(1) #Remove Mcd2 - Afternoon and evening menu
        elif hr >= 12:
            stall_names.pop(0) #Remove Mcd1 - Morning Menu

        weekdays = ("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday") #Days in a week
        day = weekdays[self.Calender.selectedDate().dayOfWeek() - 1] #Finding the day selected in the calendar

        if day == "Saturday" or day=="Sunday":
            stall_names.remove("MiniWok1") #Remove Weekday Menu
        else:
            stall_names.remove("MiniWok2") #Remove Weekend Menu

        cnt = 1
        menutext,pricetext = "",""
```

7. **dispinfo():** Function to display stall information and to check if the stall is open or closed at a given time.



```python
def dispinfo(self,text): #To display the store info and return true or false if store is open or close

    hr = self.hourcombobox.currentIndex() #Collecting data from GUI
    mint = self.minutecombobox.currentIndex()
    time = hr * 100 + mint

    store = self.tabWidget.tabText(self.tabWidget.currentIndex()) #Storing the tab text in store (Eg: McDonlads, Pizza Hut)

    f = open("stall_info.txt", "r").readlines() #Opens and reads file containing store info
    info = f[f.index(store+"\n") + 1 : f.index(store[0:2] + "\n")] #Shortens the list to the stall info


    msg = QMessageBox() #For pop up messages
    msg.setWindowTitle(store+" store info") #To set the title to the popup
    msg.setStandardButtons(QMessageBox.Ok) #Creating the OK button in pop-up window
    msg.setDetailedText(" ".join(info)) #Additional information to store info

    weekdays = ("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday") #Tuple for Week
    weekday = ("Monday","Tuesday","Wednesday","Thursday","Friday") #Tuple for Weekday

    o_c = eval(open("stall_timing.txt").read()) #Reading the file which contain opening and closing timing as dictionary

    day = weekdays[self.Calender.selectedDate().dayOfWeek()-1]
    day1 = weekdays[self.Calender.selectedDate().dayOfWeek()-1] #Getting the day of date selected in the calender

    weekday_open = o_c[store]["Weekday"][0] #Weekday opening time
    sat_open = o_c[store]["Saturday"][0] #Saturday opening time
    sun_open = o_c[store]["Sunday"][0] #Sunday opening time

    if day in weekday:
        day="Weekday" #If the day is a weekday

    if o_c[store][day][0] <= time < o_c[store][day][1]: #To check if store is open at give time
        msg.setIcon(QMessageBox.Information) #Information icon
        msg.setText("The store is OPEN for your access")
        msg.setInformativeText("Enjoy your meal! :)")

        self.Waitingtime.setEnabled(True) #Enabling the waiting time frame
        self.Waitingtime.setToolTip("Enter number of people to calculate waiting time") #Tool tip to direct to enter value
        bol = True
```

DIVISION OF LABOR

Agnesh Ramesh	Ankitha Anil	Chandrasekhar Aditya
Integrated the front end and back end together	Created Graphical User Interface using PyQt	Created back end to store stall information and menu
Created a function to display the information of menu in the front-end interface	Created a function to calculate the waiting time for a person depending on the number of people in front	Created functions to retrieve data from the files and return the filtered information depending on stall selected
Created a digital clock which constantly updates it's time	Established the connectivity between different windows	Created a function to display the current date and time

REFLECTIONS

Difficulties encountered	Solution
No prior experience in coding in Python and creating GUI using PyQt.	Learnt using PyQt documentation provided and online materials.
Indentation errors.	Debugged the error by running small bits of code separately.
Running GUI's on different laptops caused change is alignment.	Setting a specific resolution to all laptops.
Redundant statements.	Usage of functions reduced the number of lines.
Storing data in the form of data frames.	Storing the data in excel sheets

Names	Learning Outcomes
Agnesh Ramesh	The importance of commenting, indentation and restructuring the program to improve modularity and readability.
Anil Ankitha	The range of designing possibilities using PyQt and basics of GUI programming.
Chandrasekhar Aditya	Retrieval, reading and writing of data stored in text files using pandas and using datetime module.

The improvements that could be done include:

1. Adding a review system- With the usage of TextEdit and check boxes feature from PyQt, user can input reviews, and this information could be stored using MySQL database.
2. Gathering location of user- With the help of Google Maps Direction API, direction to the North Spine Food Court from the user's present location can be displayed.