CZ1003 Introduction to Computational Thinking

Mini Project

Report

Tutorial Group: FS9

Names:

He Hefei (U1922184L)

Ho Wei Yi Stanley (U1921998A)

Jabir Shah Halith (U1921464C)

Table of Contents

1 Introduction

This project report is written to showcase the features of a Real-Time NTU Canteen Information System program built using Graphical User Interface (GUI) Programming in Python. This program is developed using the PyQt5 library, where parts of the GUI are generated using the in-built Qt Designer application, as well as additional program features developed using the qtawesome library. SQLite is used to store the database for the program.

2 Objectives and Overview

The program is developed to include the following functions:

1. Store and display stall information

2. Store and display stall menus

3. Display stall information and menus based on current system date and time

4. Display stall information and menus based on user defined date and time

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

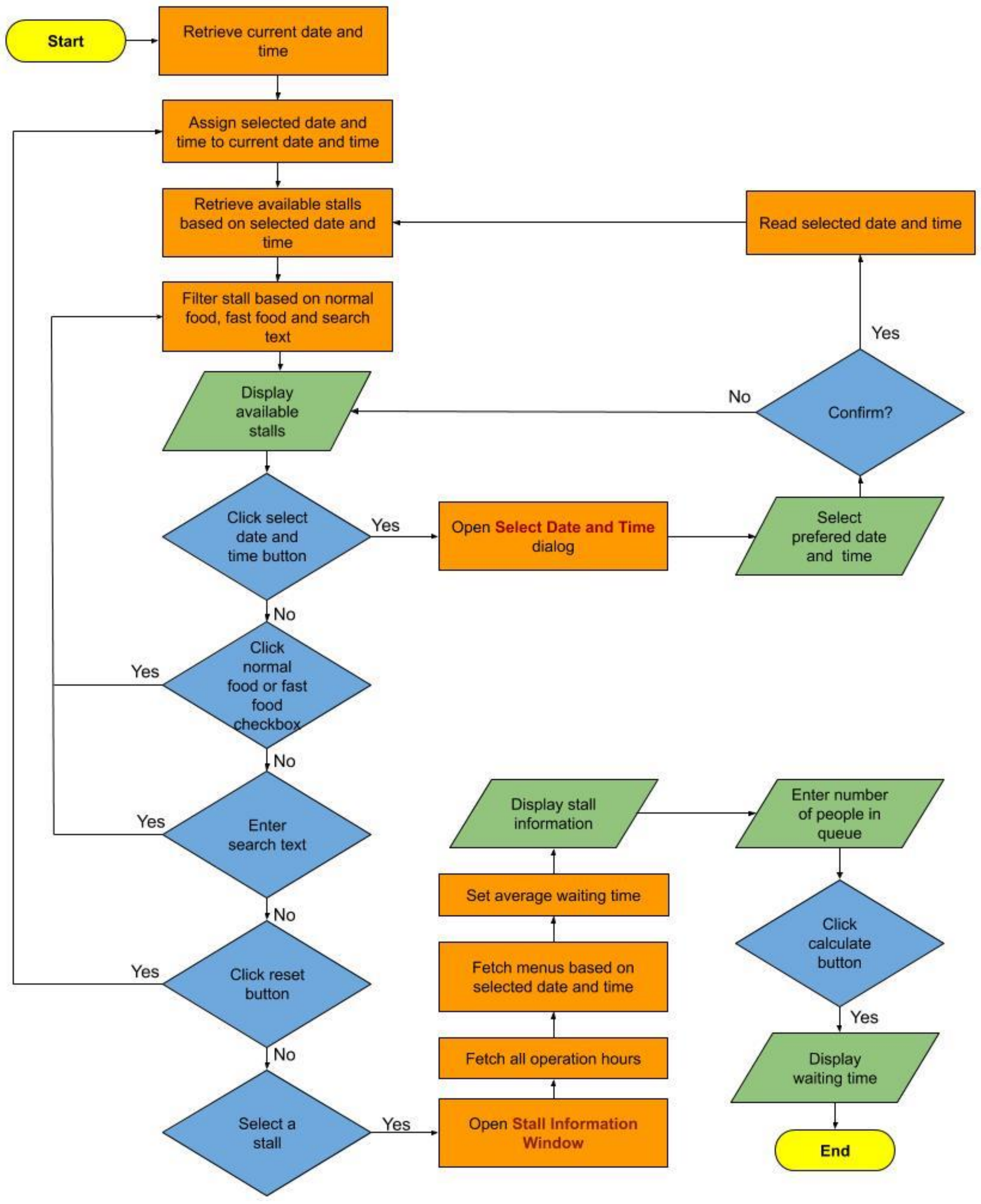
6. Allow to check the operating hours for all stalls

The python modules are split according to the format where each file contain the specific module and its functions:

- canteen_modal.py: database functions to retrieve from database and convert into various data types
- db.py: database query and opening database
- main_ui.py: UI file for main window and its functions
- mainWindowController.py: central file to hold data for all windows
- SelectDateTimeDialog.py: UI file and functions to obtain user-defined date and time
- Stall_Info_Page.py: UI file and functions to display stall information, menus and calculate waiting time

3 Algorithm Design

3.1 Top level flowchart



Start
Retrieve current date and time
Assign selected date and time to current date and time
Retrieve available stalls based on selected date and time
Read selected date and time
Filter stall based on normal food, fast food and search text
Yes
Display available stalls
No
Confirm?
Click select date and time button
Yes
Open Select Date and Time dialog
Select prefered date and time
No
Click normal food or fast food checkbox
Yes
No
Enter search text
Yes
Display stall information
Enter number of people in queue
Set average waiting time
No
Click reset button
Yes
Fetch menus based on selected date and time
Click calculate button
Yes
Fetch all operation hours
Display waiting time
No
Select a stall
Yes
Open Stall Information Window
End

Figure 1: Overall flowchart of program

3.2 User Defined Functions

3.2.1 Main Page

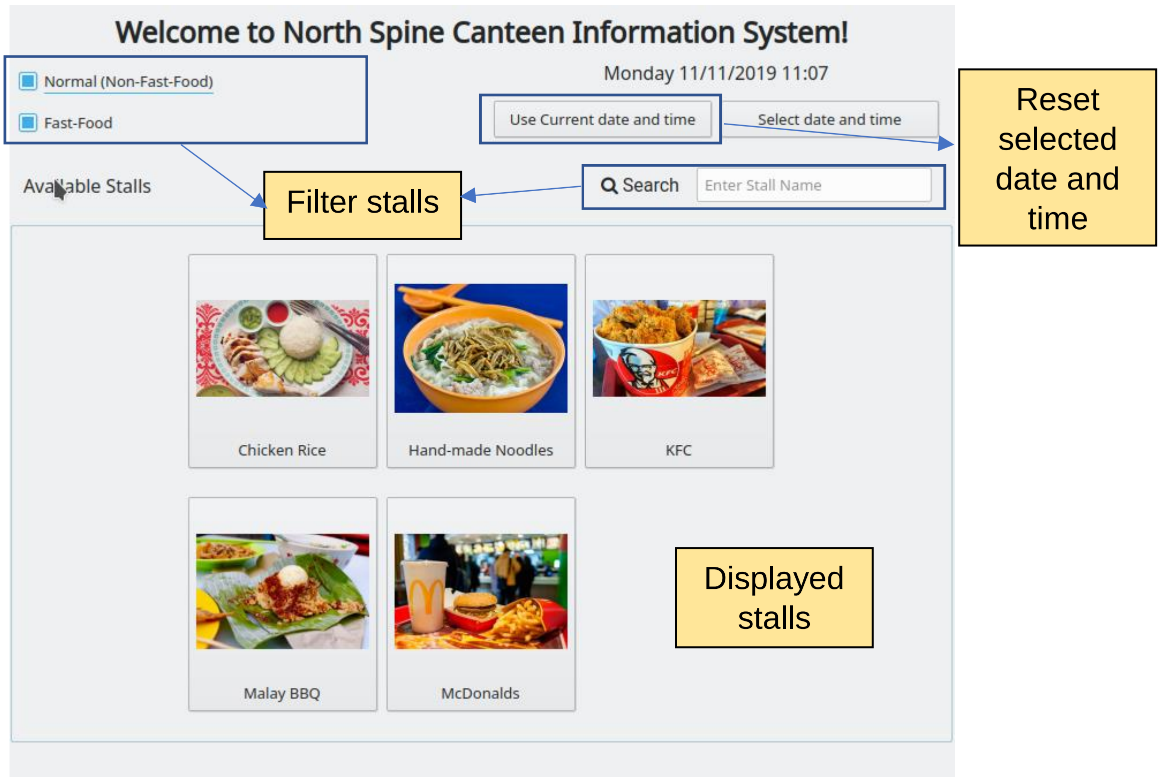
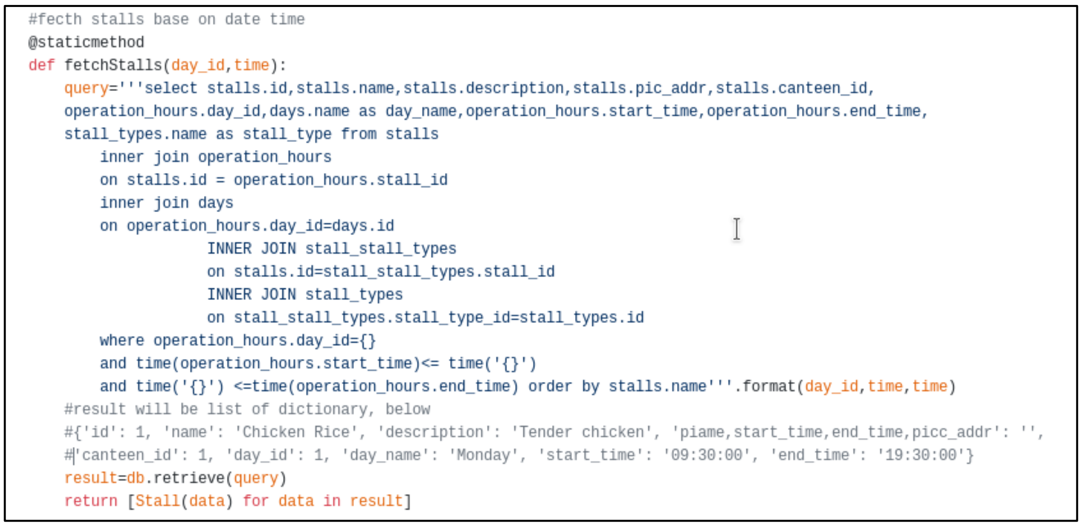


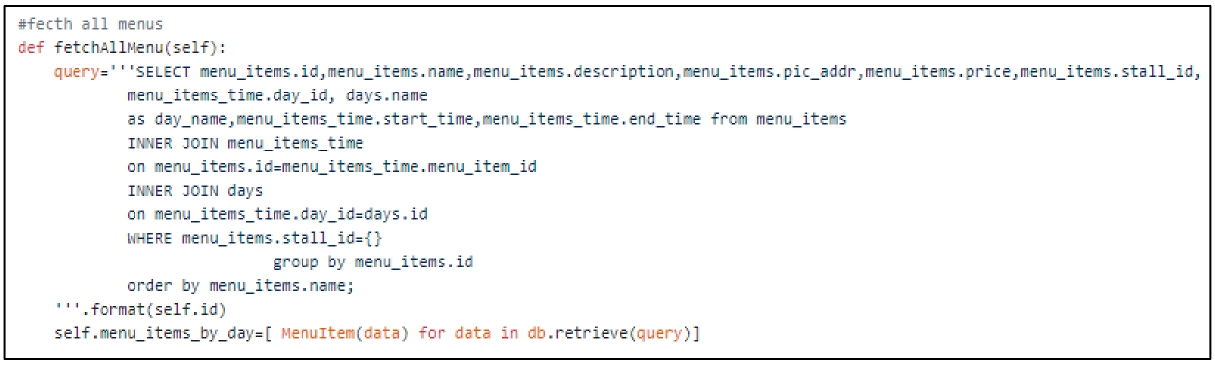
Figure 2: Main Page

Retrieve stall information and menu
Stall information and menu are retrieved from the database by the selected date and time and stored as 2 separate dictionaries.



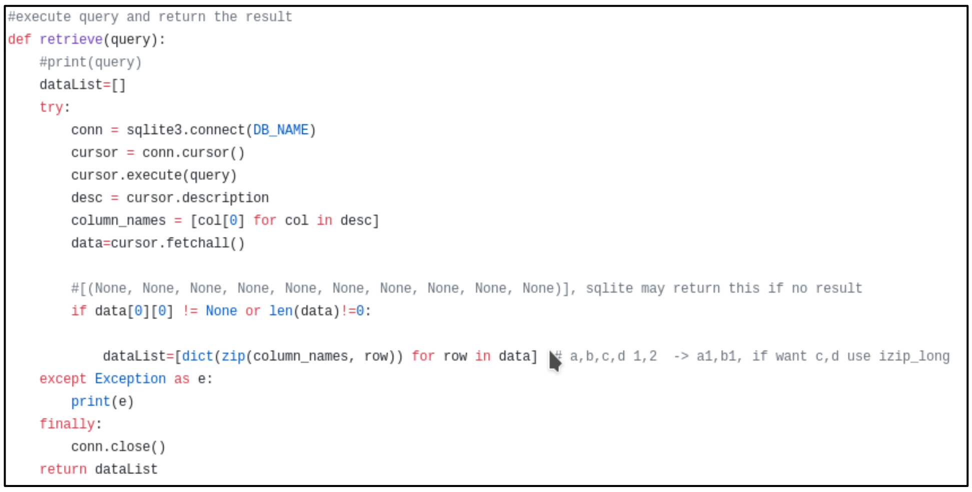
```
#fecth stalls base on date time
@staticmethod
def fetchStalls(day_id,time):
    query='''select stalls.id,stalls.name,stalls.description,stalls.pic_addr,stalls.canteen_id,
    operation_hours.day_id,days.name as day_name,operation_hours.start_time,operation_hours.end_time,
    stall_types.name as stall_type from stalls
        inner join operation_hours
        on stalls.id = operation_hours.stall_id
        inner join days
        on operation_hours.day_id=days.id
                INNER JOIN stall_stall_types
                on stalls.id=stall_stall_types.stall_id
                INNER JOIN stall_types
                on stall_stall_types.stall_type_id=stall_types.id
        where operation_hours.day_id={}
        and time(operation_hours.start_time)<= time('{}')
        and time('{}') <=time(operation_hours.end_time) order by stalls.name'''.format(day_id,time,time)
    #result will be list of dictionary, below
    #{'id': 1, 'name': 'Chicken Rice', 'description': 'Tender chicken', 'piame,start_time,end_time,picc_addr': '',
    #'canteen_id': 1, 'day_id': 1, 'day_name': 'Monday', 'start_time': '09:30:00', 'end_time': '19:30:00'}
    result=db.retrieve(query)
    return [Stall(data) for data in result]
```

Codes to retrieve stall information



```
#fecth all menus
def fetchAllMenu(self):
    query='''SELECT menu_items.id,menu_items.name,menu_items.description,menu_items.pic_addr,menu_items.price,menu_items.stall_id,
            menu_items_time.day_id, days.name
            as day_name,menu_items_time.start_time,menu_items_time.end_time from menu_items
            INNER JOIN menu_items_time
            on menu_items.id=menu_items_time.menu_item_id
            INNER JOIN days
            on menu_items_time.day_id=days.id
            WHERE menu_items.stall_id={}
                            group by menu_items.id
            order by menu_items.name;
    '''.format(self.id)
    self.menu_items_by_day=[ MenuItem(data) for data in db.retrieve(query)]
```

Codes to retrieve stall menus



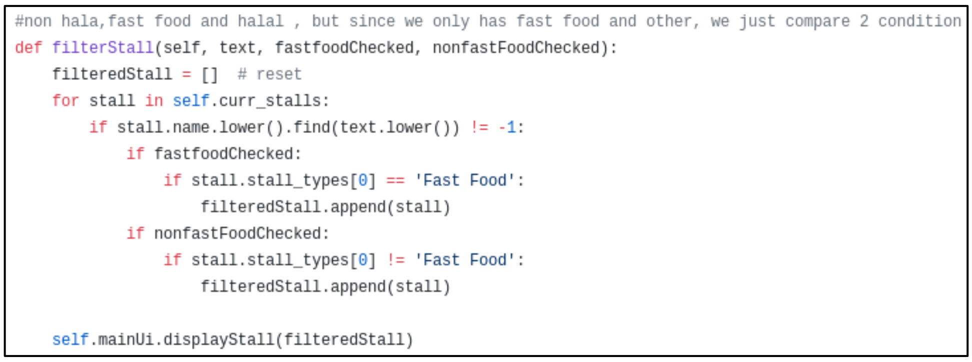
```
#execute query and return the result
def retrieve(query):
    #print(query)
    dataList=[]
    try:
        conn = sqlite3.connect(DB_NAME)
        cursor = conn.cursor()
        cursor.execute(query)
        desc = cursor.description
        column_names = [col[0] for col in desc]
        data=cursor.fetchall()

        #[(None, None, None, None, None, None, None, None, None, None)], sqlite may return this if no result
        if data[0][0] != None or len(data)!=0:

            dataList=[dict(zip(column_names, row)) for row in data]  # a,b,c,d 1,2  -> a1,b1, if want c,d use izip_long
    except Exception as e:
        print(e)
    finally:
        conn.close()
    return dataList
```

Codes to retrieve selected information above to store as dictionaries

Filter stall

This method will filter the stall based on 3 criteria:

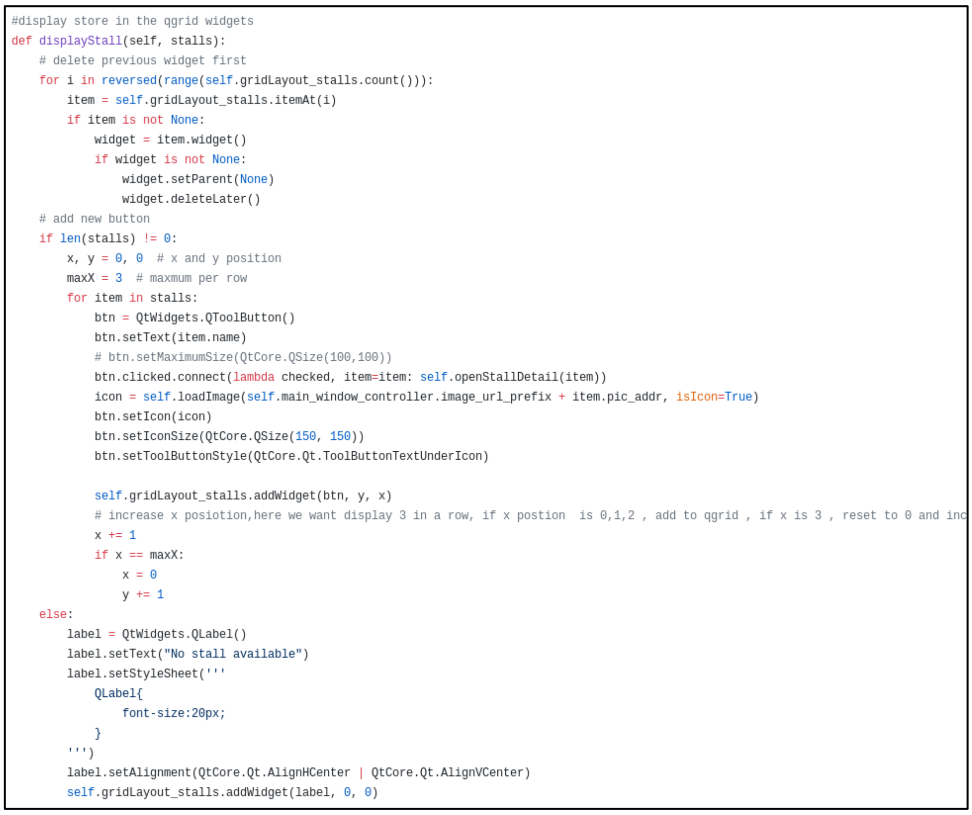
- State of Normal checkbox
- State of Fast-food checkbox
- The search text that user has entered



```python
#non hala,fast food and halal , but since we only has fast food and other, we just compare 2 condition
def filterStall(self, text, fastfoodChecked, nonfastFoodChecked):
    filteredStall = []  # reset
    for stall in self.curr_stalls:
        if stall.name.lower().find(text.lower()) != -1:
            if fastfoodChecked:
                if stall.stall_types[0] == 'Fast Food':
                    filteredStall.append(stall)
            if nonfastFoodChecked:
                if stall.stall_types[0] != 'Fast Food':
                    filteredStall.append(stall)

    self.mainUi.displayStall(filteredStall)
```

Display stall

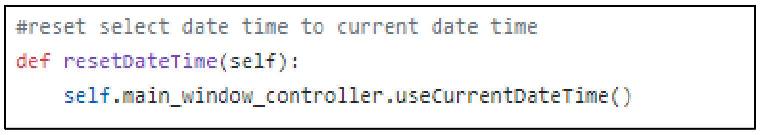
This function will accept a list of stalls as parameter and display it in the UI.



```python
#display store in the qgrid widgets
def displayStall(self, stalls):
    # delete previous widget first
    for i in reversed(range(self.gridLayout_stalls.count())):
        item = self.gridLayout_stalls.itemAt(i)
        if item is not None:
            widget = item.widget()
            if widget is not None:
                widget.setParent(None)
                widget.deleteLater()
    # add new button
    if len(stalls) != 0:
        x, y = 0, 0  # x and y position
        maxX = 3  # maxmum per row
        for item in stalls:
            btn = QtWidgets.QToolButton()
            btn.setText(item.name)
            # btn.setMaximumSize(QtCore.QSize(100,100))
            btn.clicked.connect(lambda checked, item=item: self.openStallDetail(item))
            icon = self.loadImage(self.main_window_controller.image_url_prefix + item.pic_addr, isIcon=True)
            btn.setIcon(icon)
            btn.setIconSize(QtCore.QSize(150, 150))
            btn.setToolButtonStyle(QtCore.Qt.ToolButtonTextUnderIcon)

            self.gridLayout_stalls.addWidget(btn, y, x)
            # increase x posiotion,here we want display 3 in a row, if x postion  is 0,1,2 , add to qgrid , if x is 3 , reset to 0 and inc
            x += 1
            if x == maxX:
                x = 0
                y += 1
    else:
        label = QtWidgets.QLabel()
        label.setText("No stall available")
        label.setStyleSheet('''
            QLabel{
                font-size:20px;
            }
        ''')
        label.setAlignment(QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter)
        self.gridLayout_stalls.addWidget(label, 0, 0)
```

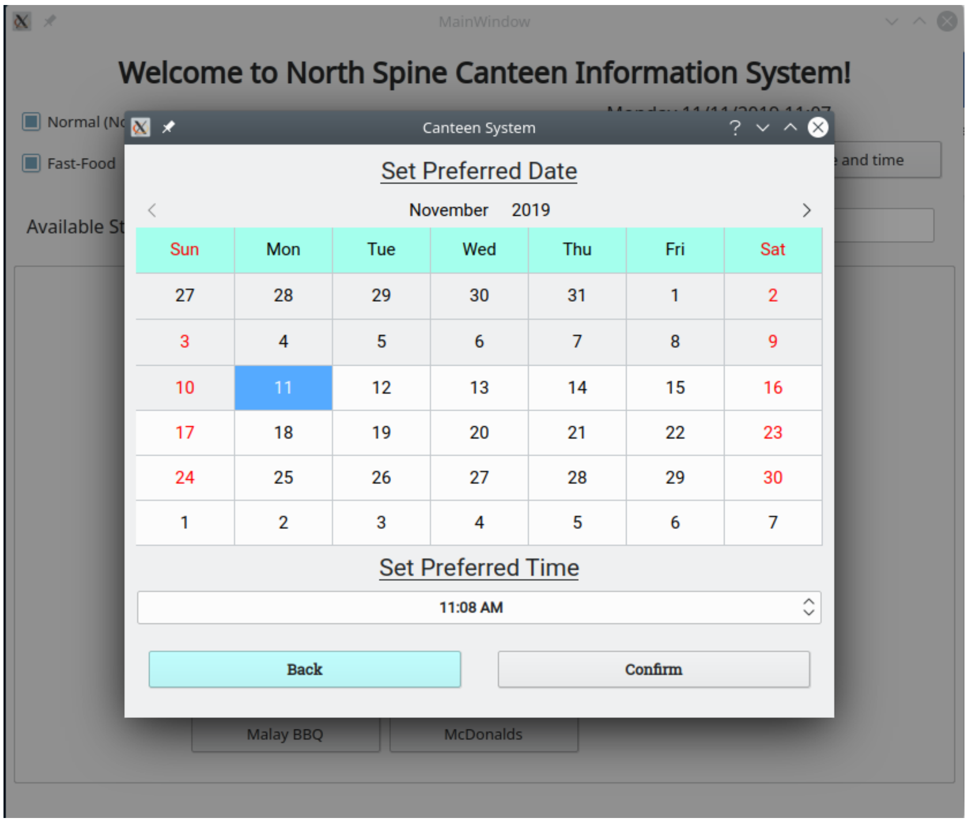
Reset selected date and time

This function sets the system’s selected date and time to current date and time.



```python
#reset select date time to current date time
def resetDateTime(self):
    self.main_window_controller.useCurrentDateTime()
```

3.2.2 Select Date and Time Page

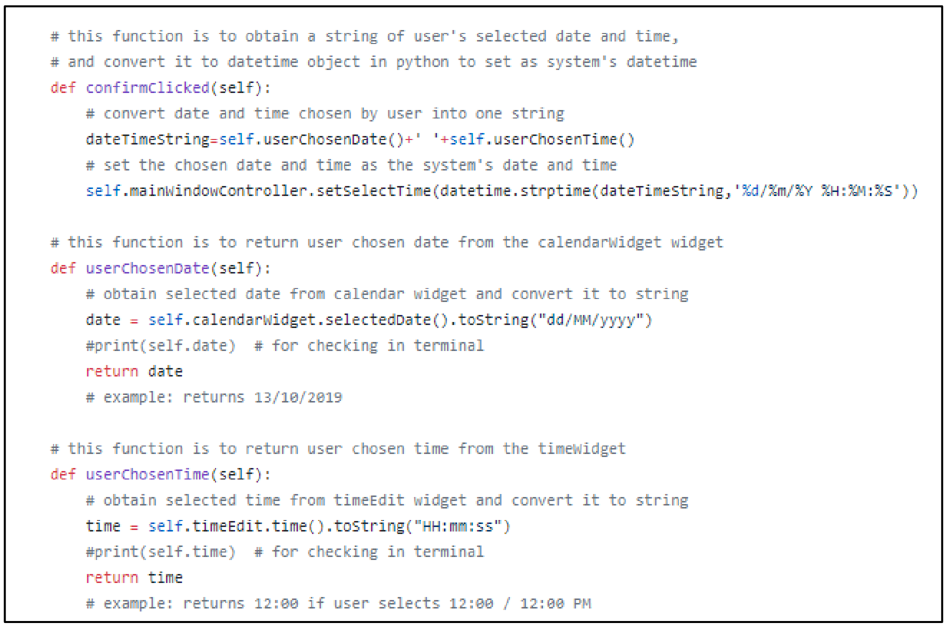


Sun	Mon	Tue	Wed	Thu	Fri	Sat
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

Figure 3: Select Date and Time page

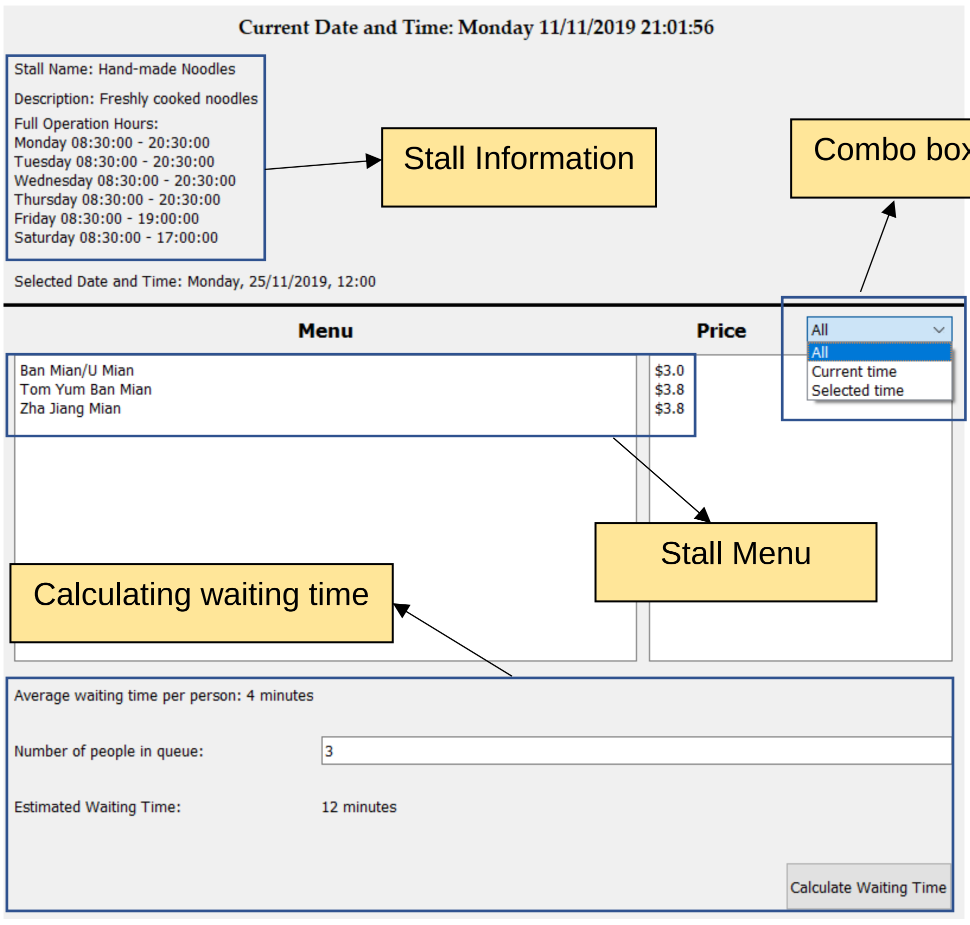
Choose date and time

The calendar widget and timeEdit widget allows user to select their preferred date and time, where this selected information will be sent to the main window controller module to store this timing for use in displaying stalls and menus from selected date and time.



```python
# this function is to obtain a string of user's selected date and time,
# and convert it to datetime object in python to set as system's datetime
def confirmClicked(self):
    # convert date and time chosen by user into one string
    dateTimeString=self.userChosenDate()+' '+self.userChosenTime()
    # set the chosen date and time as the system's date and time
    self.mainWindowController.setSelectTime(datetime.strptime(dateTimeString,'%d/%m/%Y %H:%M:%S'))

# this function is to return user chosen date from the calendarWidget widget
def userChosenDate(self):
    # obtain selected date from calendar widget and convert it to string
    date = self.calendarWidget.selectedDate().toString("dd/MM/yyyy")
    #print(self.date)  # for checking in terminal
    return date
    # example: returns 13/10/2019

# this function is to return user chosen time from the timeWidget
def userChosenTime(self):
    # obtain selected time from timeEdit widget and convert it to string
    time = self.timeEdit.time().toString("HH:mm:ss")
    #print(self.time)  # for checking in terminal
    return time
    # example: returns 12:00 if user selects 12:00 / 12:00 PM
```

3.2.3 Stall Information Page



Current Date and Time: Monday 11/11/2019 21:01:56

Stall Name: Hand-made Noodles

Description: Freshly cooked noodles

Full Operation Hours:
Monday 08:30:00 - 20:30:00
Tuesday 08:30:00 - 20:30:00
Wednesday 08:30:00 - 20:30:00
Thursday 08:30:00 - 20:30:00
Friday 08:30:00 - 19:00:00
Saturday 08:30:00 - 17:00:00

Selected Date and Time: Monday, 25/11/2019, 12:00

All
- All
- Current time
- Selected time

Menu	Price
Ban Mian/U Mian	$3.0
Tom Yum Ban Mian	$3.8
Zha Jiang Mian	$3.8

Average waiting time per person: 4 minutes

Number of people in queue: 3

Estimated Waiting Time: 12 minutes

Calculate Waiting Time

Figure 4: Stall Information Page

Display Stall Information

From selected stall, stall information (including operating hours) printed on the UI are retrieved from the dictionary created in ‘Retrieve Stall Information’.

```
#open stall detail and pass main window controller
def openStallDetail(self,stall):
    self.Stall_Info_Window = QtWidgets.QMainWindow()
    self.Stall_Info_Window.setWindowModality(QtCore.Qt.ApplicationModal)
    self.stallDetail = Ui_Stall_Info_Window()
    self.stallDetail.setupUi(self.Stall_Info_Window,stall,self.main_window_controller)
    self.Stall_Info_Window.show()
```

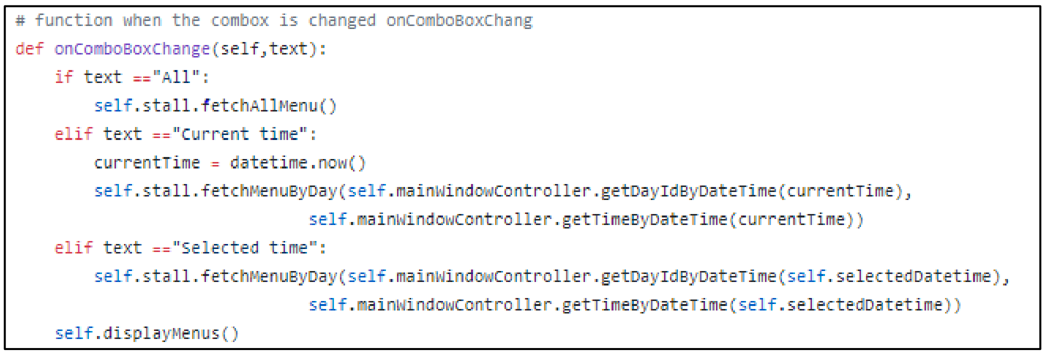
Codes to connect Main window with Stall Information Window

```
def retranslateUi(self, Stall_Info_Window):
    _translate = QtCore.QCoreApplication.translate

    # fetch Stall information from canteen_modal, { Stall name, Stall Description, Operating Hours & Selected Date and Time to respective labels }
    self.label_stallName.setText(_translate("Stall_Info_Window", "Stall Name: " + self.stall.name))
    self.label_stallDesc.setText(_translate("Stall_Info_Window", "Description: " + self.stall.description))
    self.label_stallOpHour.setText(
        _translate("Stall_Info_Window", "Full Operation Hours:\n" + self.stall.getAllOperationHoursInString()))
```

Codes to display stall information

Combo Box Change

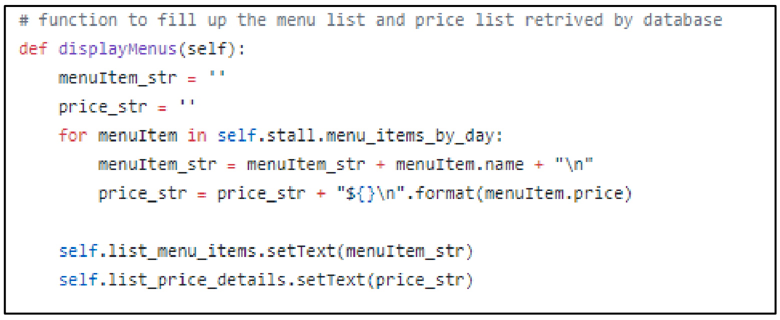
The combo box change function is to select menu by user’s selected time, current time or all menu items.



```python
# function when the combox is changed onComboBoxChang
def onComboBoxChange(self,text):
    if text =="All":
        self.stall.fetchAllMenu()
    elif text =="Current time":
        currentTime = datetime.now()
        self.stall.fetchMenuByDay(self.mainWindowController.getDayIdByDateTime(currentTime),
                              self.mainWindowController.getTimeByDateTime(currentTime))
    elif text =="Selected time":
        self.stall.fetchMenuByDay(self.mainWindowController.getDayIdByDateTime(self.selectedDatetime),
                              self.mainWindowController.getTimeByDateTime(self.selectedDatetime))
    self.displayMenus()
```

Display Menu

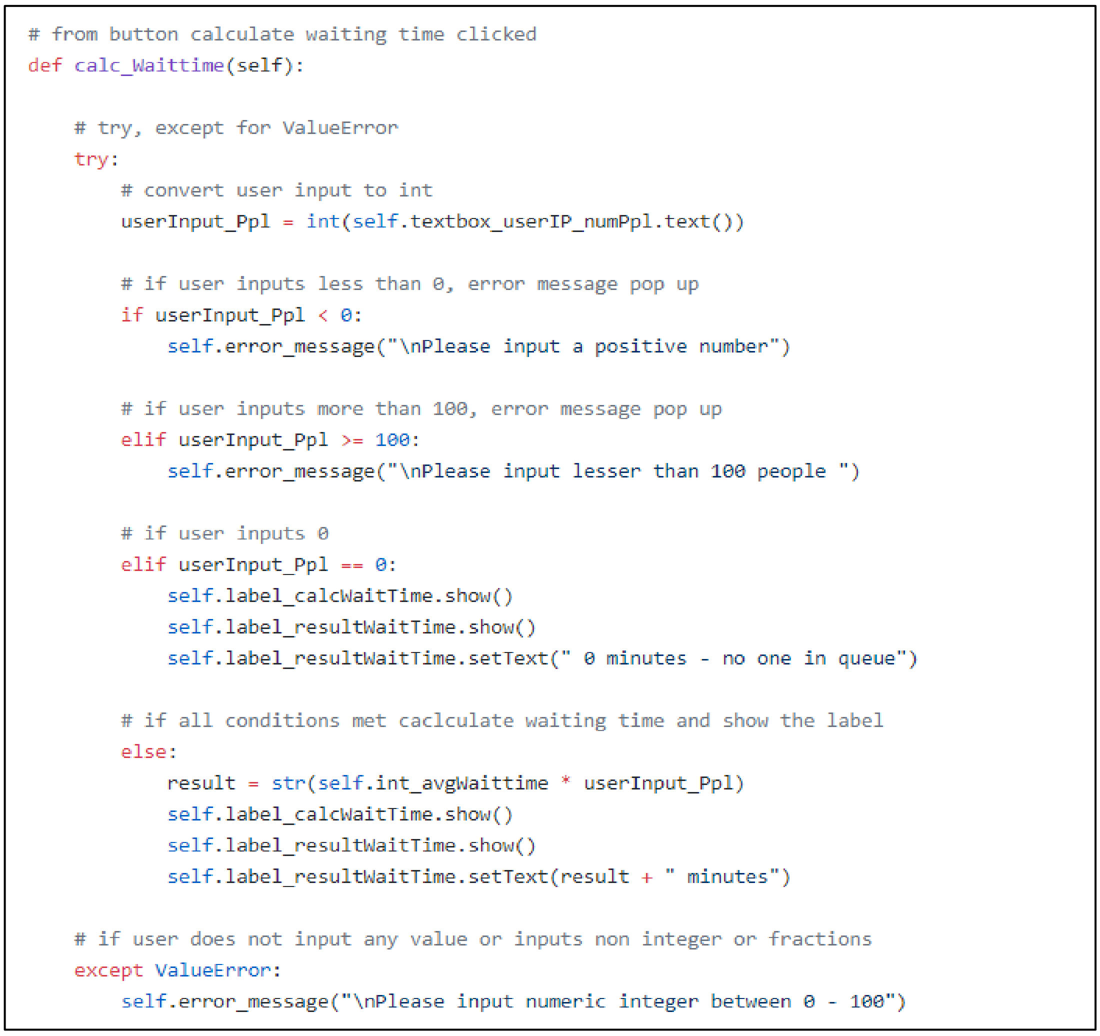
With reference from the combo box, the menu and the price will be printed on the UI.



```python
# function to fill up the menu list and price list retrived by database
def displayMenus(self):
    menuItem_str = ''
    price_str = ''
    for menuItem in self.stall.menu_items_by_day:
        menuItem_str = menuItem_str + menuItem.name + "\n"
        price_str = price_str + "${}\n".format(menuItem.price)

    self.list_menu_items.setText(menuItem_str)
    self.list_price_details.setText(price_str)
```

Calculating Waiting Time

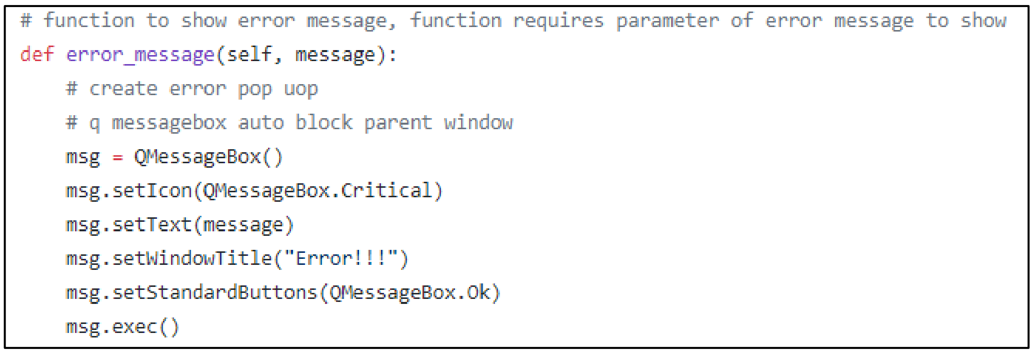
This function will be invoked when user clicks the calculate button. It checks for error and respond based on the validity of the input. The detailed error checking can be found in the program testing page.



```python
# from button calculate waiting time clicked
def calc_Waittime(self):

    # try, except for ValueError
    try:
        # convert user input to int
        userInput_Ppl = int(self.textbox_userIP_numPpl.text())

        # if user inputs less than 0, error message pop up
        if userInput_Ppl < 0:
            self.error_message("\nPlease input a positive number")

        # if user inputs more than 100, error message pop up
        elif userInput_Ppl >= 100:
            self.error_message("\nPlease input lesser than 100 people ")

        # if user inputs 0
        elif userInput_Ppl == 0:
            self.label_calcWaitTime.show()
            self.label_resultWaitTime.show()
            self.label_resultWaitTime.setText(" 0 minutes - no one in queue")

        # if all conditions met caclculate waiting time and show the label
        else:
            result = str(self.int_avgWaittime * userInput_Ppl)
            self.label_calcWaitTime.show()
            self.label_resultWaitTime.show()
            self.label_resultWaitTime.setText(result + " minutes")

    # if user does not input any value or inputs non integer or fractions
    except ValueError:
        self.error_message("\nPlease input numeric integer between 0 - 100")
```

Error Message

With reference from the above function, when an error is detected, the Error Message will pop up. Valid inputs are required from users.



```python
# function to show error message, function requires parameter of error message to show
def error_message(self, message):
    # create error pop uop
    # q messagebox auto block parent window
    msg = QMessageBox()
    msg.setIcon(QMessageBox.Critical)
    msg.setText(message)
    msg.setWindowTitle("Error!!!")
    msg.setStandardButtons(QMessageBox.Ok)
    msg.exec()
```

4 Program Testing

4.1 Main Page

4.1.1 Search Function

Prerequisite: Normal and Fast food checkbox checked, select any date that falls on Monday and 12:00 pm, all 5 stalls should be displayed.

No.	Description	Steps	Test Data	Expected Results	Actual Result	Pass /Fail
1	Find KFC	Enter text in the search text field	1. Kfc 2. KFC 3. kfc	Only KFC stall displayed	Only KFC stall displayed	1. Pass 2. Pass 3. Pass
2	Stalls		1. 89ji 2. ;';[	Display "no stall available"	Display "no stall available"	1. Pass 2. Pass
3	Display Stall contain character 'k'		1. K 2. k	Display Chicken rice and KFC	Display Chicken rice and KFC	1. Pass 2. Pass
4	Display all 5 stores if search text field is empty	Delete all characters in text field	Empty search text field	Display 5 stalls	Display 5 stalls	Pass

Figure 5: Testing of search function

4.1.2 Non-fast (Normal) food and fast-food checkbox

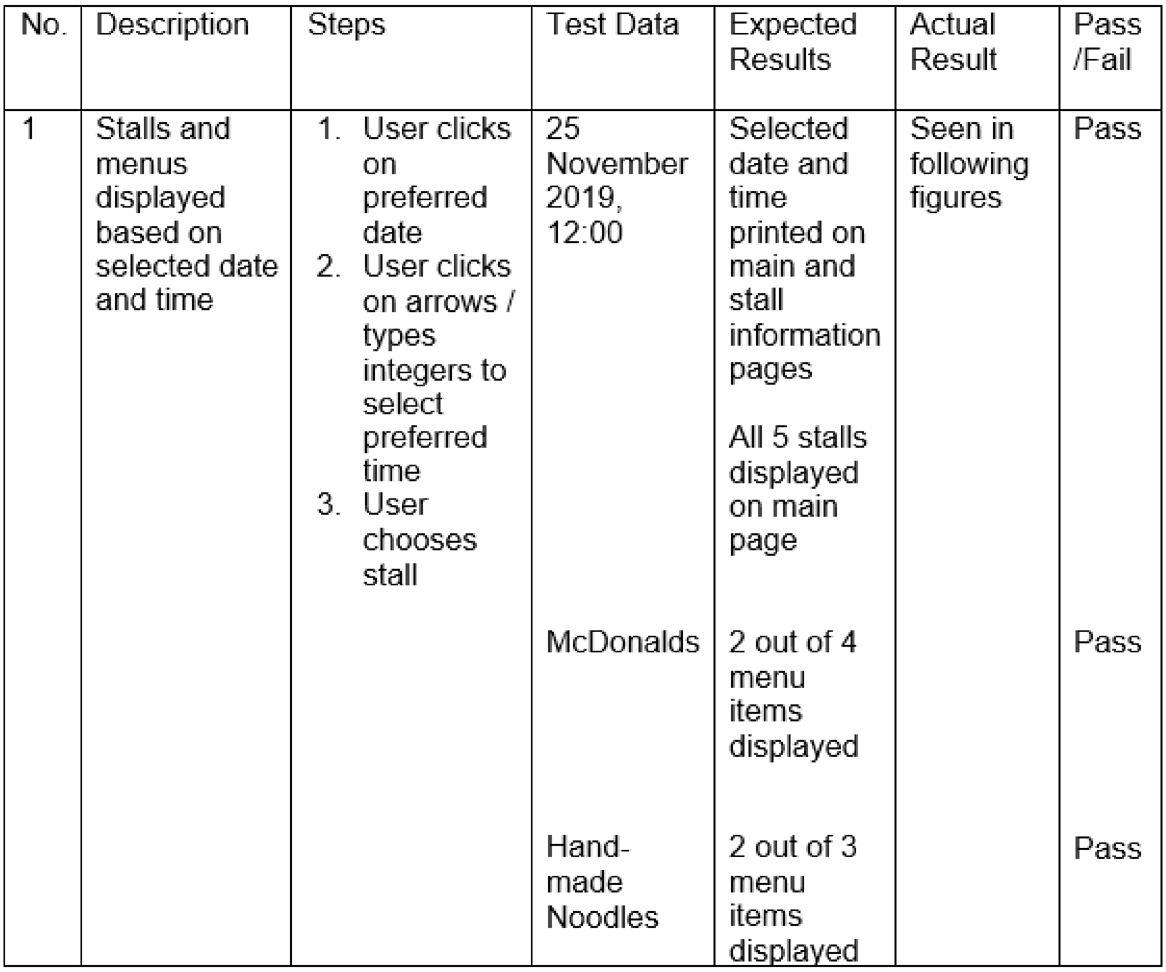
Prerequisite: select any date that falls on Monday and 12:00 pm, all 5 stalls should be displayed, and search box left empty

No.	Description	Steps	Expected Results	Actual Result	Pass /Fail
1	Display both normal and fast food	Checked both normal and fast-food checkbox	Total 5 stalls display	Total 5 stalls display	Pass
2	Display Normal food	Only Normal checkbox checked	Display Chicken rice hand-made noodle and Malay BBQ	Display Chicken rice hand-made noodle and Malay BBQ	Pass
3	Display fast food	Only fast-food checkbox checked	Display McDonald and KFC	Display McDonald and KFC	Pass
4	Display no Stall	Uncheck both normal and fast-food checkbox	Display "no stall available"	Display "no stall available"	Pass

Figure 6: Testing of checkbox function

4.2 Date and Time Page

The use of the in-built PyQt5 calendar and timeEdit widgets ensure that users are unable to input invalid values such as non-integers or out-of-range integers, hence eliminating the need to check for the validity of inputs. Default values of current date and time will be selected for cases where users did not change inputs.



No.	Description	Steps	Test Data	Expected Results	Actual Result	Pass /Fail
1	Stalls and menus displayed based on selected date and time	1. User clicks on preferred date 2. User clicks on arrows / types integers to select preferred time 3. User chooses stall	25 November 2019, 12:00	Selected date and time printed on main and stall information pages All 5 stalls displayed on main page	Seen in following figures	Pass
			McDonalds	2 out of 4 menu items displayed		Pass
			Hand-made Noodles	2 out of 3 menu items displayed		Pass

Figure 7: Testing of obtaining user-defined date and time function

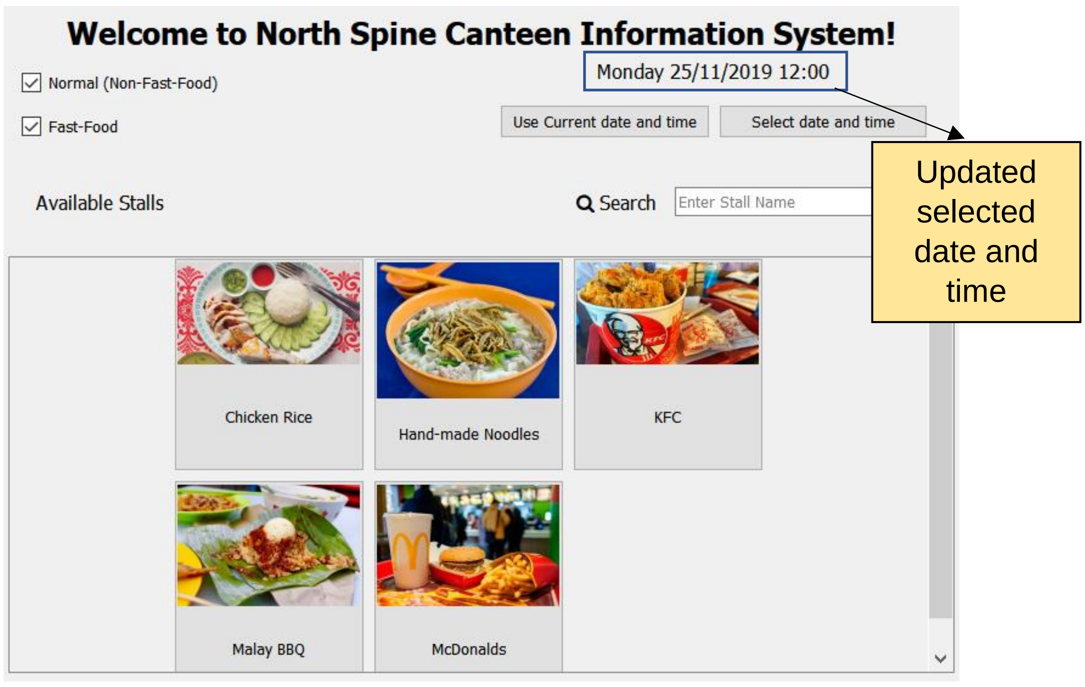
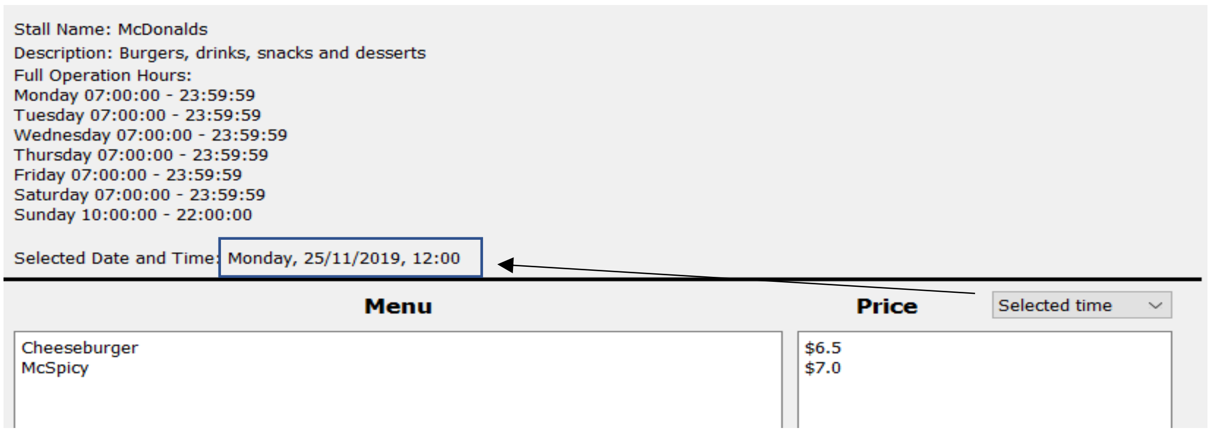


Figure 8: Updated Main Page

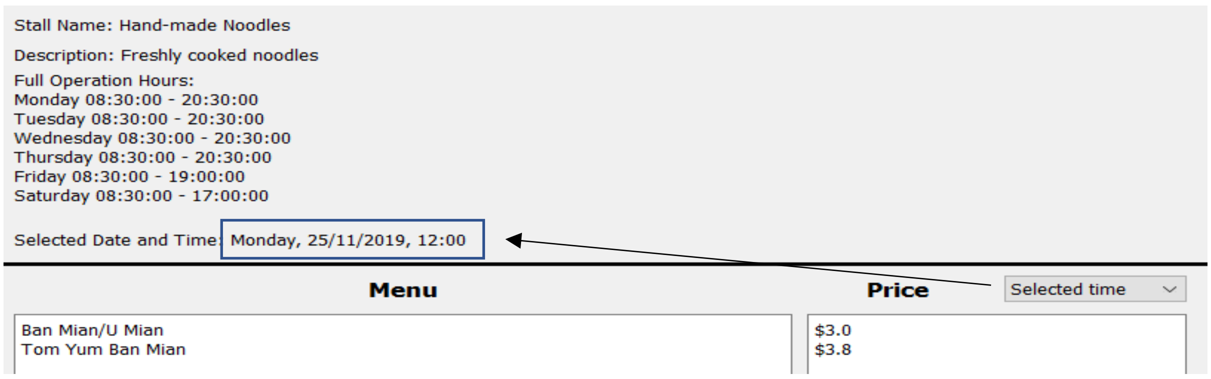


Stall Name: McDonalds
Description: Burgers, drinks, snacks and desserts
Full Operation Hours:
Monday 07:00:00 - 23:59:59
Tuesday 07:00:00 - 23:59:59
Wednesday 07:00:00 - 23:59:59
Thursday 07:00:00 - 23:59:59
Friday 07:00:00 - 23:59:59
Saturday 07:00:00 - 23:59:59
Sunday 10:00:00 - 22:00:00

Selected Date and Time: Monday, 25/11/2019, 12:00

Menu	Price
Cheeseburger	$6.5
McSpicy	$7.0

Figure 9: Menu from McDonalds



Stall Name: Hand-made Noodles
Description: Freshly cooked noodles
Full Operation Hours:
Monday 08:30:00 - 20:30:00
Tuesday 08:30:00 - 20:30:00
Wednesday 08:30:00 - 20:30:00
Thursday 08:30:00 - 20:30:00
Friday 08:30:00 - 19:00:00
Saturday 08:30:00 - 17:00:00

Selected Date and Time: Monday, 25/11/2019, 12:00

Menu	Price
Ban Mian/U Mian	$3.0
Tom Yum Ban Mian	$3.8

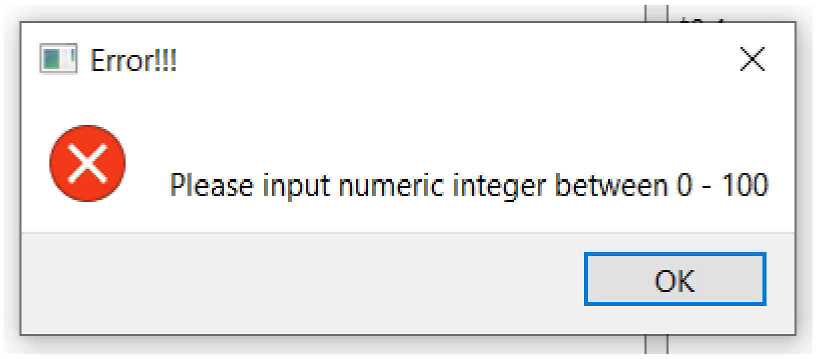
Figure 10: Menu from Hand-made Noodles

4.3 Stall Information Page

Users have the option to calculate estimated waiting time based on the number of people in the queue. This gives rise to error cases when they do not input valid integers. These cases are described in the following figure.

No.	Description	Steps	Test Data	Expected Results	Actual Result	Pass /Fail
1	Maximum Input Check	1: Enter test data into the text field 2: Click on calculate estimated waiting time button	999999	Error	Popup Error Message	Pass
2	Float Input Checking		5.0			Pass
3	Alphabetical Characters Input Check		A abcd			Pass
4	Null Input Check		-blank-space			Pass
5	Negative Input Check		-21			Pass
6	Special Character Input Check		* @			Pass

Figure 11: Testing of user inputs in text field



Error!!!

Please input numeric integer between 0 - 100

Figure 12: Popup Error Message (different message for different invalid inputs)

5 Reflections

5.1 Hefei

The mini project was a great opportunity for me to practice what I have learned in class. Apart from these, I also learned how to construct python class, common widgets, layout, and QSS in PyQt5, make query to sqlite3 and use git to integrate teammate's code. The most challenging part was not familiar with PyQt5 framework, most of the time the code does not work as intended, thus need a lot of research and reading on documentation. Furthermore, I have helpful teammates who helped me overcome these obstacles from the discussions I had with them.

5.2 Stanley

Having no prior experience to programming, building this mini project with cooperative and experienced teammates helped me learn aplenty. Learning PyQt5's library was very challenging. I faced many problems initially when my codes did not work as intended due to not using the correct functions and parameters. To overcome such problems, I added print statements as I code to check for desired outputs and used the "Debug" function in PyCharm to point out which lines caused errors. Being tasked to input information into our database using SQLite meant I had to learn SQL commands that the system accepts. We also added more comments to our codes for greater readability and learned to communicate more to help each other understand our codes. Lastly, learning how to share my codes through Github brought me an insight that this is an efficient way to merge our codes.

5.3 Shah

One of the greatest difficulties I faced was being able to adapt to the python environment. Python had rather a different syntax and it took some time to learn the new syntax. I managed to find some YouTube videos and along with the knowledge in lab sessions, I managed to overcome this difficulty to adapt. Doing projects in a group to program was challenging as I had only experience working alone previously. I have managed to overcome it as I had great teammates who were cooperative. I had difficulties in learning how to call a function. Practicing in the lab and project have made it easier to understand. Since we were using Github, I managed to learn how to effectively code as a group using Github. Using PyQt Designer was a challenging task as I need to keep a backup because every time the UI was updated, my codes will be erased. I learnt how to use PyCharm to resolve this as it was easier to use. These were my learning points in this project.

6 Conclusion

This Real-Time Canteen Information System program allows users to conveniently check for specific stalls and its menus and information based on current or user-defined date and time. It can be further improved by allowing the system to check for statuses of stalls during public holidays.

7 References

'Chicken Rice' image: https://imageresizer.static9.net.au/OBxyNbouafFnEdnGRNygyL-3dac=/600x338/smart/https%3A%2F%2Fprod.static9.net.au%2F_%2Fmedia%2F2017%2F05%2F26%2F15%2F55%2FPohs-Hainanese-chicken-rice.jpg

'Hand-made Noodles' image: http://img.insing.com/FoodDrink/T&T-banmian-qiurong.jpg

'KFC' image: https://ichef.bbci.co.uk/news/660/cpsprodpb/66A2/production/_107847262_gettyimages-1152276563-594x594.jpg

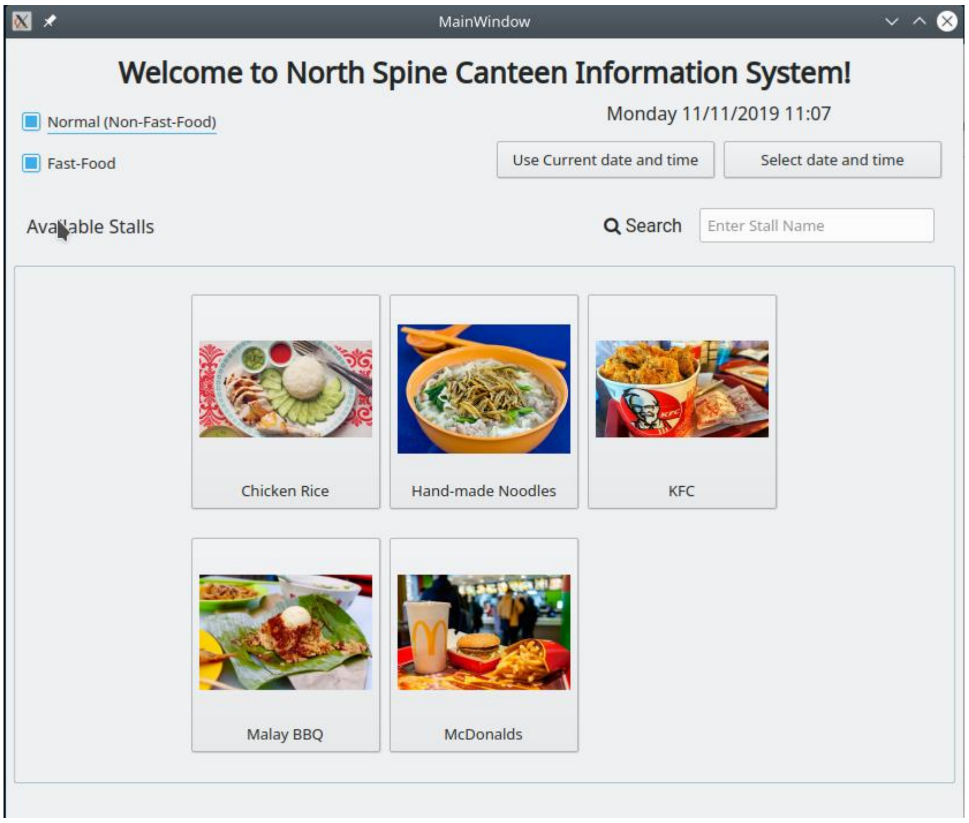
'Malay BBQ' image: https://chasingaplate.com/wp-content/uploads/2018/01/IMG_1504-1.jpeg

'McDonalds' image: https://cdn-01.independent.ie/incoming/article36646079.ece/bf867/AUTOCROP/w620/os_183906250_l_2015.jpg

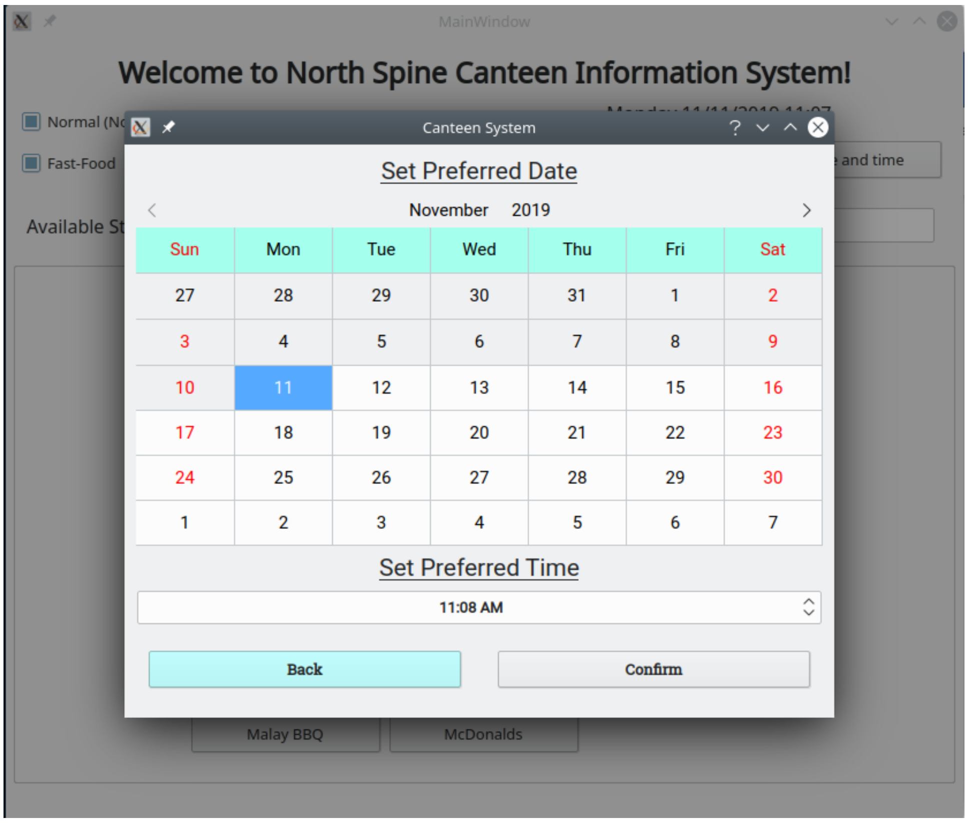
Icon image: https://cdn3.iconfinder.com/data/icons/food-155/100/Healthy_food_2-512.png

8 Appendix

Main Window (He HeFei)



Date Time Window (Stanley)



Set Preferred Date

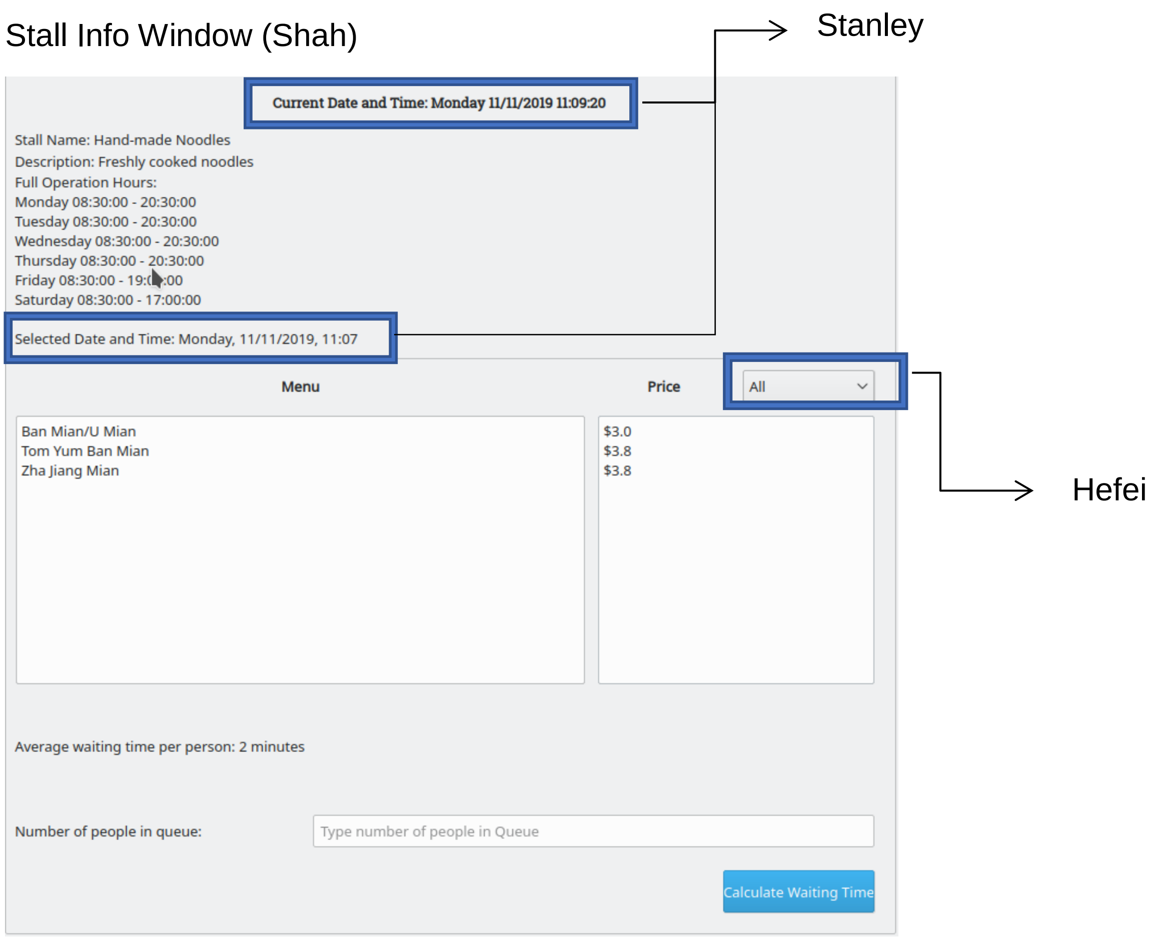
November 2019

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

Set Preferred Time

11:08 AM

Stall Info Window (Shah)



Current Date and Time: Monday 11/11/2019 11:09:20

Stall Name: Hand-made Noodles
Description: Freshly cooked noodles
Full Operation Hours:
Monday 08:30:00 - 20:30:00
Tuesday 08:30:00 - 20:30:00
Wednesday 08:30:00 - 20:30:00
Thursday 08:30:00 - 20:30:00
Friday 08:30:00 - 19:00:00
Saturday 08:30:00 - 17:00:00

Selected Date and Time: Monday, 11/11/2019, 11:07

Menu	Price
Ban Mian/U Mian	$3.0
Tom Yum Ban Mian	$3.8
Zha Jiang Mian	$3.8

Average waiting time per person: 2 minutes

Number of people in queue: Type number of people in Queue