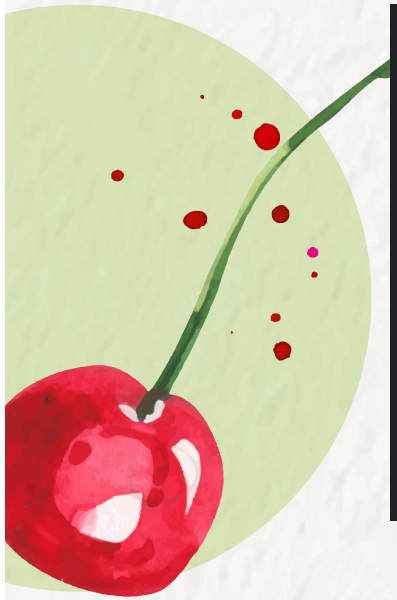
NTU CANTEEN HELPER

~ GUAT KWAN ~ CHI HUI ~ LAN

01. External Library and Modules

Modules used:

- Datetime
- PyQt
- json

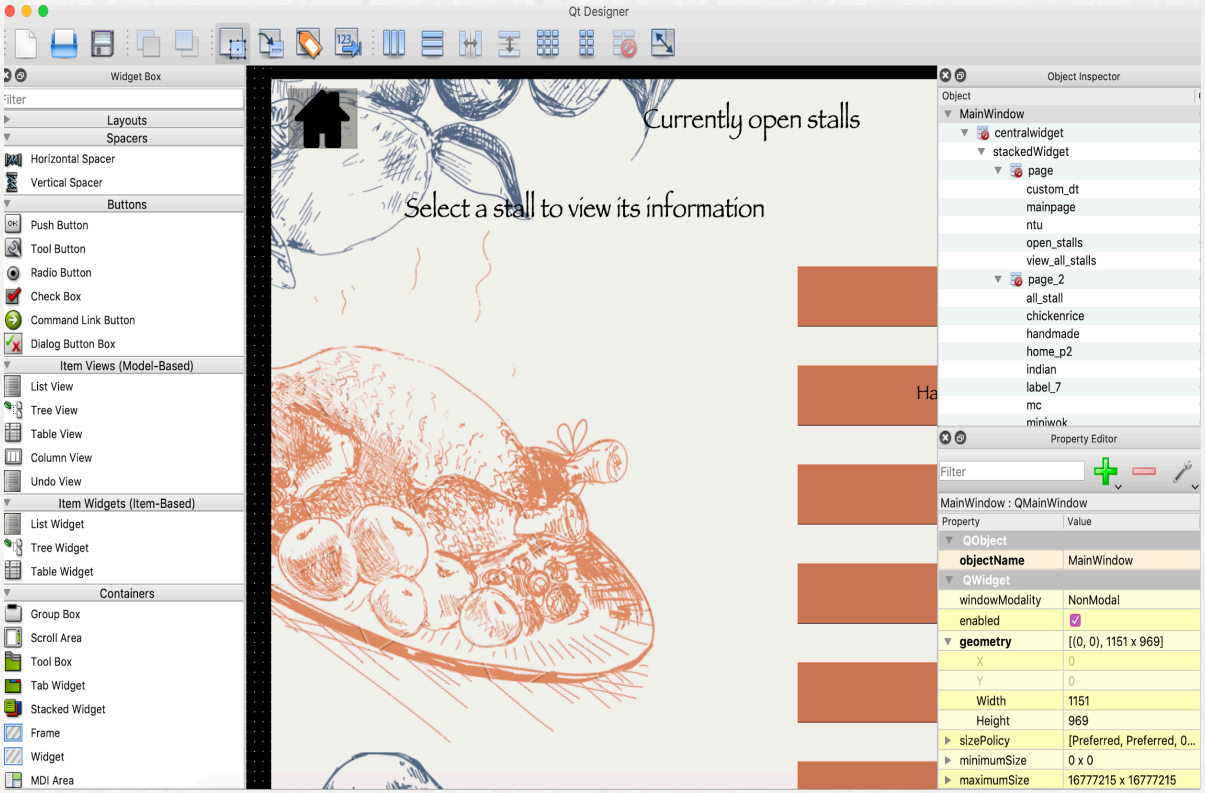


```
from datetime import datetime
from PyQt5.QtCore import QDate, QTime
from PyQt5 import QtGui
from PyQt5.QtWidgets import QMessageBox
import json
```

01. External Library and Modules

Why PyQt?

1. Simplicity
2. Ease of generating UI file using PyQt5 designer



Assessment Criteria

02.Program Organisation

Use of 5 files

- 1 UI file
  - GUI2.ui
- 2 python files
  - GUI_func.py
  - GUI_main.py
- 2 Json file
  - opr_hr_db.json
  - menu_db.json

```
from GUI_func_12nov_updated import *
import sys



class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        uic.loadUi('GUI2.ui', self)
```

03.File Operation

- Nested Dictionaries use to store data
- Databases stored as Json File

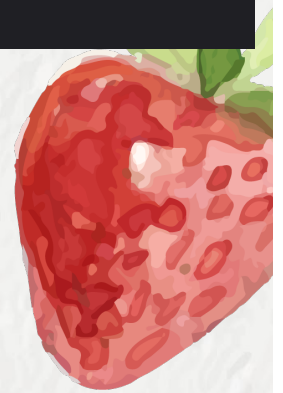
Why Json?

Simplicity of json

- Straightforward Syntax
- Easy to read and write
- Language independent

```
# import menu database
with open('menu_db.json') as json_file:
    menu_db = json.load(json_file)

# import operating hours database
with open('opr_hr_db.json') as json_file:
    opr_hr_db = json.load(json_file)
```



04. Dictionary and List

```
'McDonald\'s':
    {'Wait Time': 1,
     'Special': 'Time',
     'Special Timings':
         {'Breakfast': ['07:00:00', '11:00:00'],
          'Regular': ['11:00:01', '23:59:59']},
     'Menu':
         {'Breakfast':
             {'Big Breakfast': '$7.50',
              'Scrambled Egg Burger Set': '$6.30',
              'Breakfast Wrap Chicken Ham': '$6.20'},
          'Lunch':
             {'McSpicy Set': '$7.70',
              'Crispy Fish Sandwich Set': '$8.40',
              'Chicken McNuggets Set': '$6.70',
              'McWing': '$2.7'}}},
```

Use of list to store start and end times

Dictionary of menu

05. String Operations/Functions

```
def menu_to_string(menu):
    text_list = []
    for item, price in menu.items():
        text_list.append(str(item) + ': ' + str(price) + '\n\n')
    return ''.join(text_list)
```

String concatenation
using .join() method

06. Exception Handling

```
def wait_time_check(input, error):
    try:
        no_in_queue = int(input.toPlainText())
        if 101 > no_in_queue >= 0:
            error.clear()
            global waitTime
            wt = no_in_queue * waitTime
            error.setText('Waiting time: ' + str(wt) + ' minute(s)')
        else:
            input.clear()
            error.setText('Please enter a positive number'
                          ' within 0 and 100 inclusive.')
    except ValueError:
        error.clear()
        error.setText('Please enter an integer.')
```

Try converting user input into int

Display waiting time

Out-of-range erro

Incorrect data type error

07. Clarity and Comprehensibility

Segmentation

- Functions are placed under different categories based on their purpose
- E.g. Date and time functions, data retrieving functions, GUI functions

07. Clarity and Comprehensibility

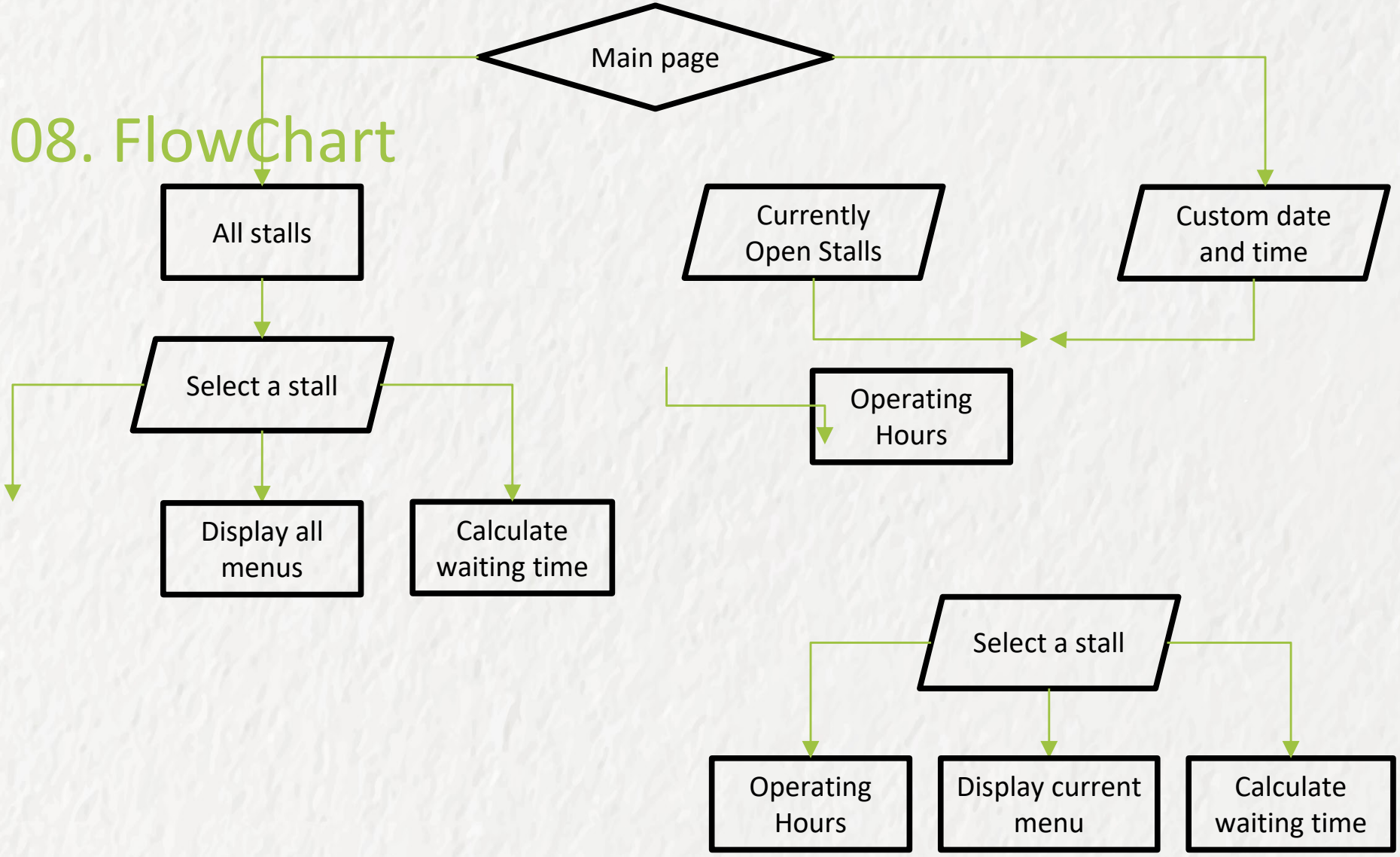
Comments and Documentation

```
# For listing all the stalls that are open currently
# input: day = the day to check for (string)
#        date_time = the time to check for (datetime)
# output: list of open stalls (list)
def find_open_stalls(day, date_time):
```

Description of inputs

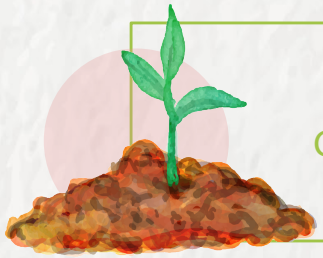
Brief description of function

08. FlowChart



09. LIVE DEMO

10. Contributions



GUAT KWAN

- Allow to check operating hours of all store
- Display informations and menu based on current date and time
- Overall Algorithm and flow of program

LAN

- Store and display stall menus
- Store and display stall information based on user defined date and time
- GUI

CHI HUI

- Store and display stall menu
- Calculate estimated waiting time for stall by asking user to enter the number of people in queue
- GUI

Q&A

THANK YOU