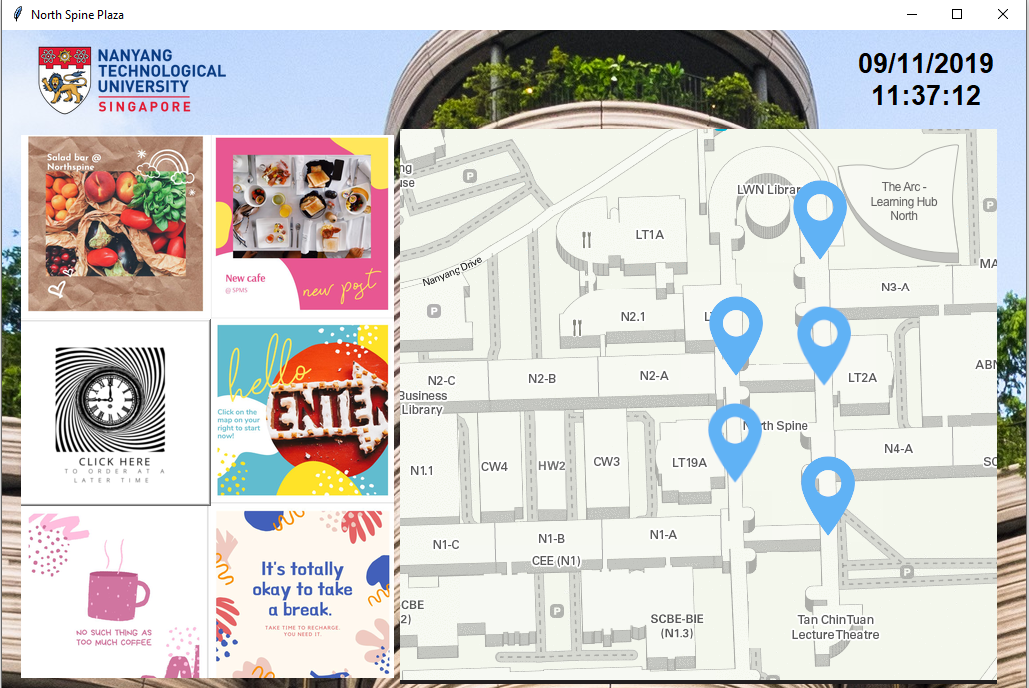
**CZ1003 Project Report**

**By Tan Jun Hong, Venkat Subramanian and Tey Chin Yi**

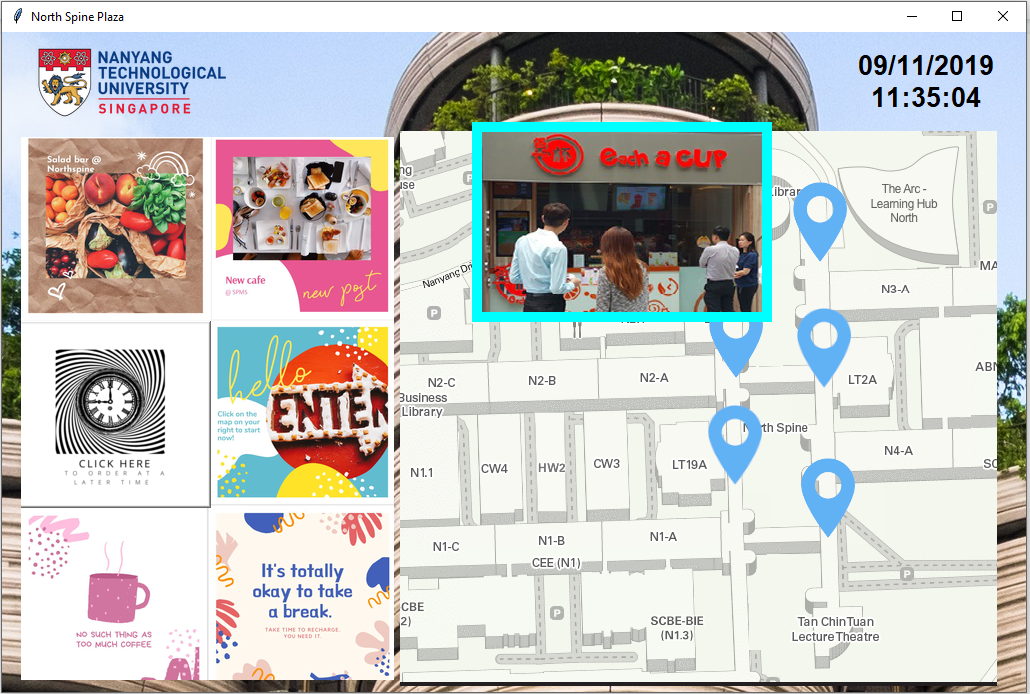
|  |  |
| --- | --- |
| **Table of content** | |
| **1** | [**Main page (Algorithm design)**](#bookmark=id.gjdgxs)  **Mainly coded by Venkat** |
| **2** | [**Second page (Algorithm design)**](#bookmark=id.30j0zll)  **Mainly coded by Chin Yi** |
| **3** | [**Individual modules (Algorithm design)**](#bookmark=id.1fob9te)  **Mainly coded by Jun Hong** |
| **4** | [**Individual functions (Algorithm design)**](#bookmark=id.3znysh7)  **Mainly coded by Jun Hong** |
| **5** | [**Error handling test cases**](#bookmark=id.2et92p0) |
| **6** | [**Individual reflection**](#bookmark=id.tyjcwt) |

**1) Main page**

* Display stalls available in North Spine. Stall image popup when mouse hover over the location.

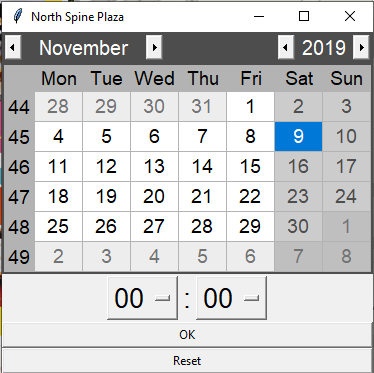


(Fig 1: Main page)



(Fig 2: Hover)

User can browse menu at a different date/time by specifying date and time via the calendar widget. Else, current date/time will be selected.



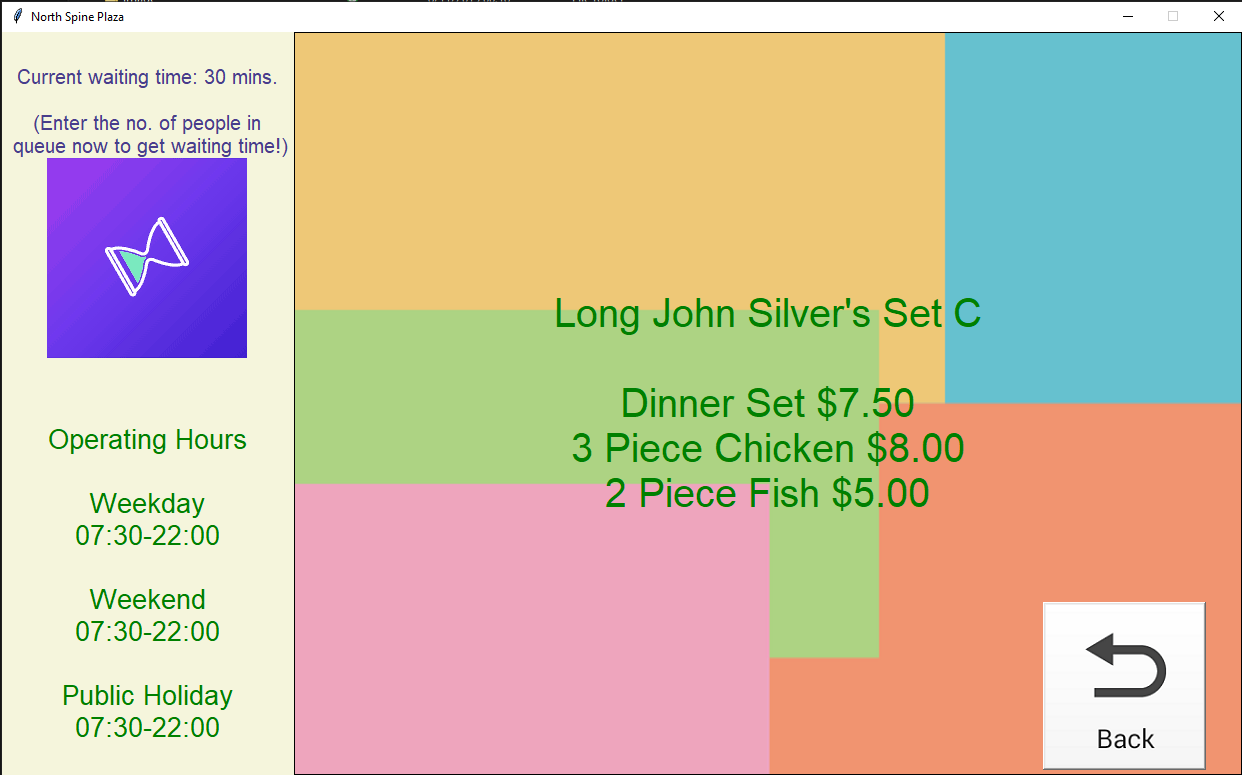
(Fig 3: Clock button) (Fig 4: Calendar widget)

* Selected time will be displayed on the main page.



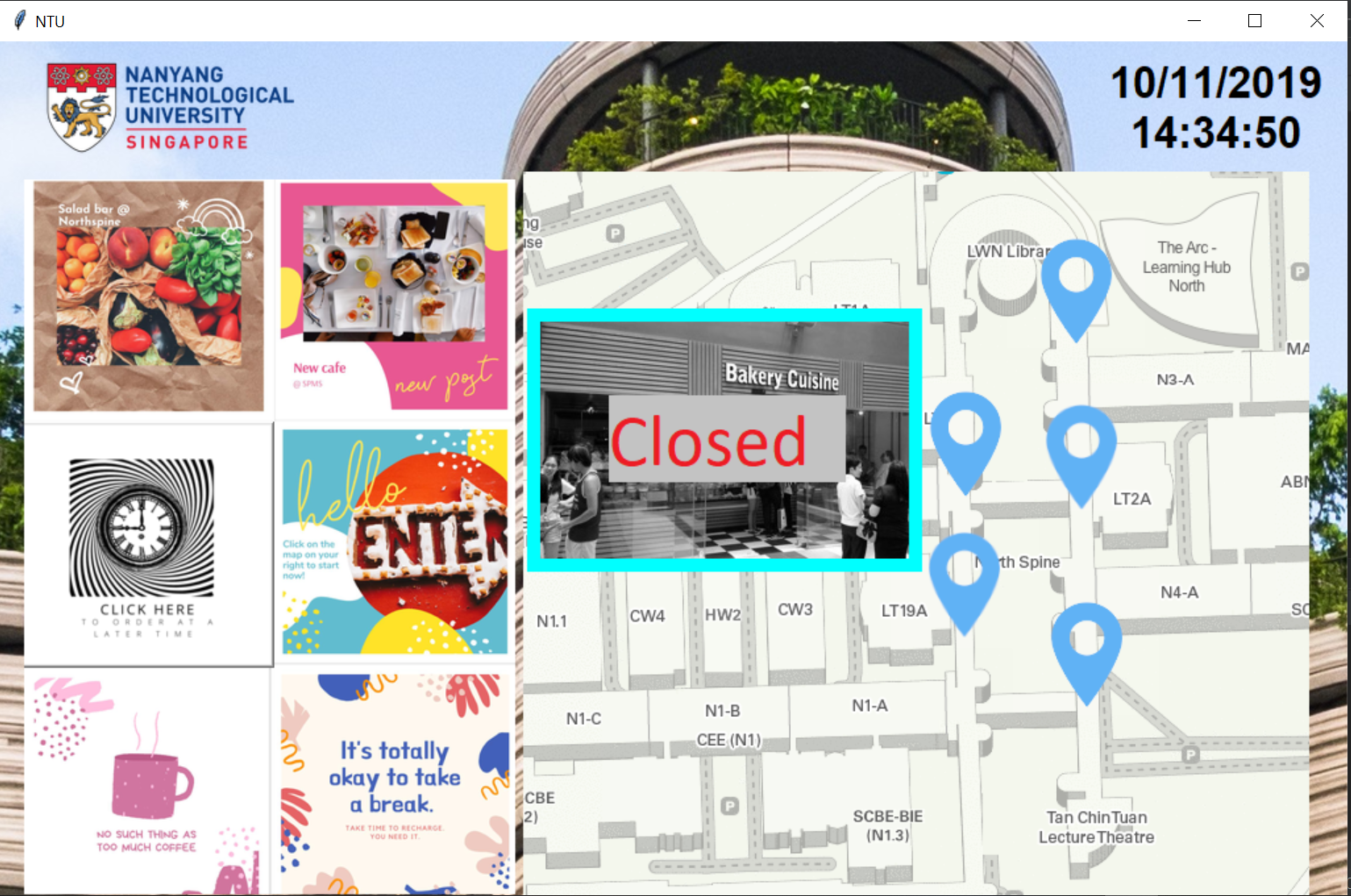
(Fig 5: Selected time and date on top of the map)

* Clicking on the location pin will show menu at selected date/time.



(Fig 6: Second page)

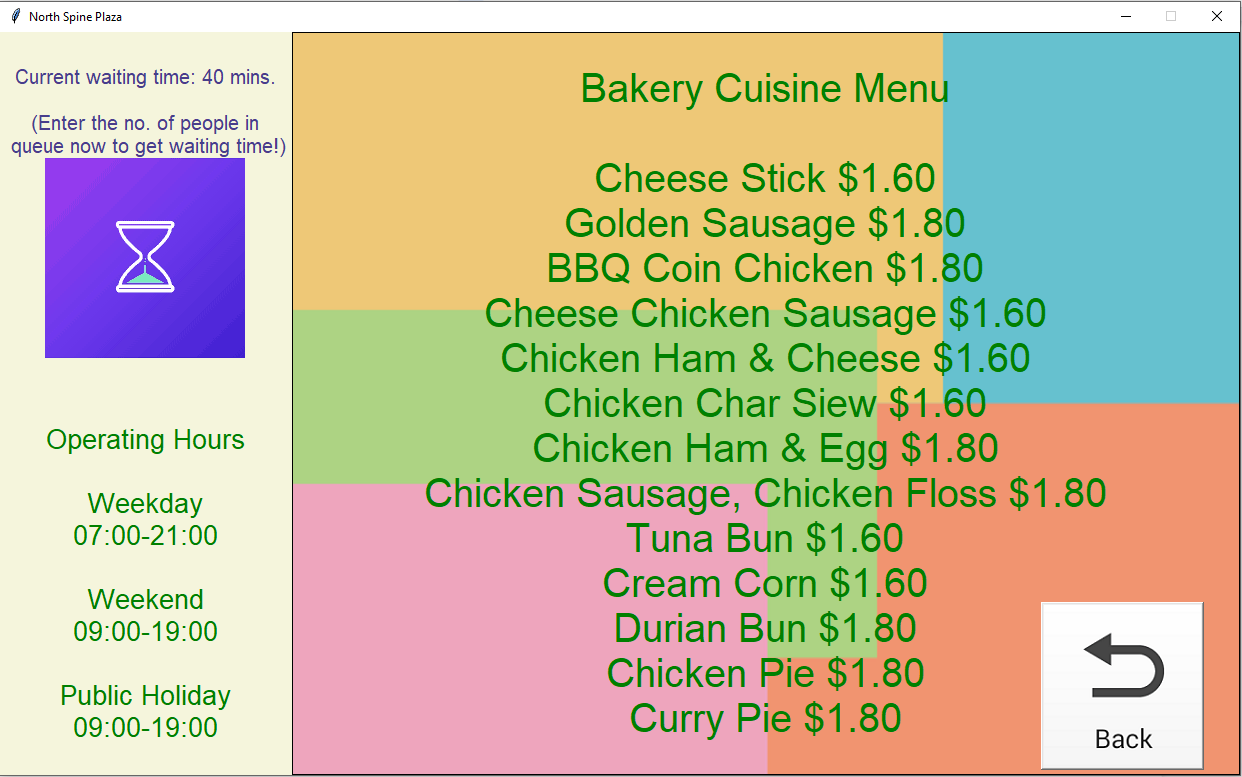
If stall is closed at the selected time, stall image would be greyed out and labelled ‘Closed’ to inform the user stall is closed.



(Fig 7: Stall closed)

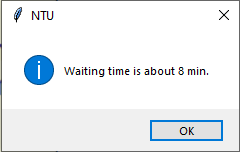
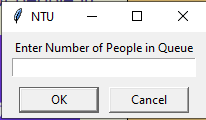
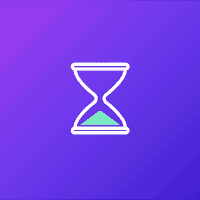
**2) Second page**

* The second page displays current waiting time (if stall is open), operating hours and the menu at the specified time.



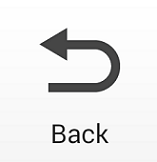
(Fig 8: Second page)

* Current (estimated) waiting time is displayed.
* The user can input the number of people to estimate new waiting time.



(Fig 9: Estimated waiting time)

* The back button helps user to get back to the first page.



(Fig 10: Back button)

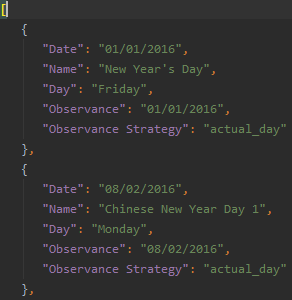
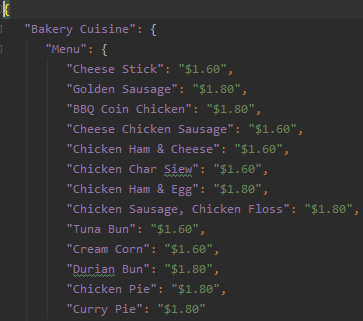
If the stall is closed at the specified time, the waiting time label and button will be disabled, with an indication that stall is closed.

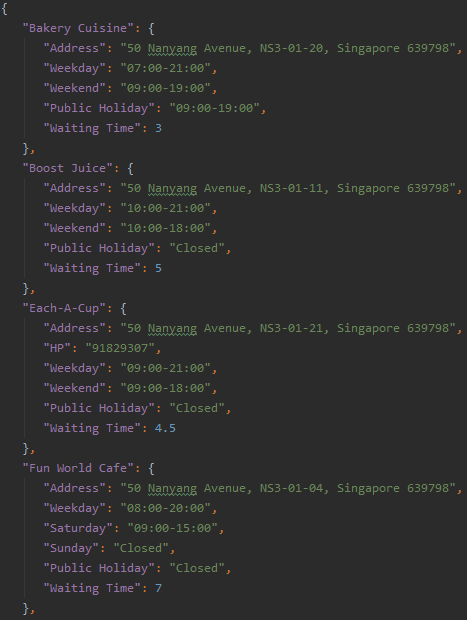


(Fig 10: Stall closed)

**3) Individual modules**

* json files to store data -- better readability and easier manipulation.
* Easier for users to add data in because of the easier readability.

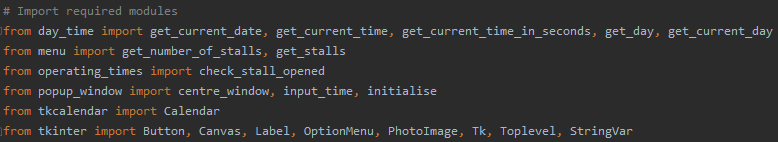




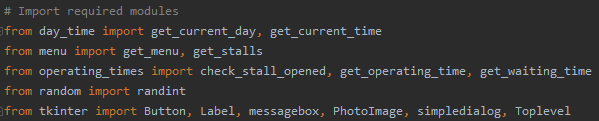
(Fig 11: json files)

Functions are separated into different modules for better organization.

* All functions have docstring and proper comments.
* Modules and functions are labelled with easy to understand names to facilitate the importing process.



(Fig 11: Importing modules for main page)



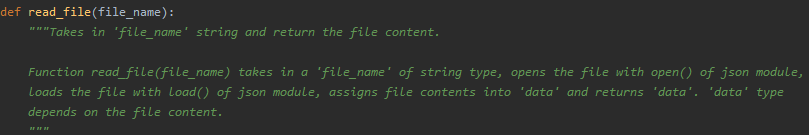
(Fig 12: Importing modules for second page)

**4) Individual functions**

* Below are some important functions:

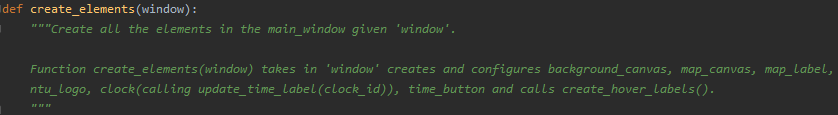
**json\_reader**

read\_file() to read json file

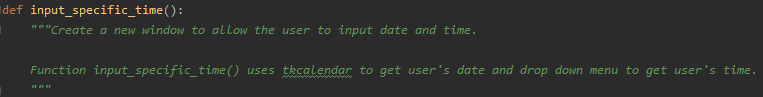


**main**

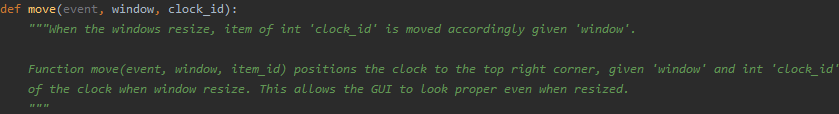
create\_elements() to initialise all the GUI elements



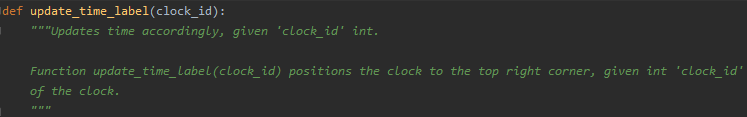
input\_specific\_time() to call a window with tkcalendar (external library), which allows the user to input date, and a drop down menu for user to input time



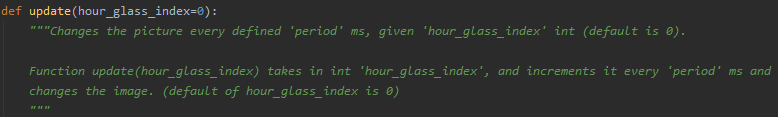
move() to move all the elements in the window when resized, so the GUI elements look properly placed at any geometry size



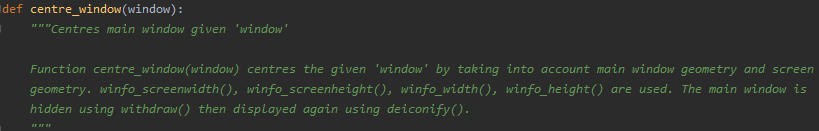
update\_time\_label() to allow the clock to run



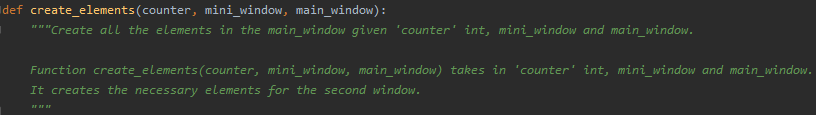
**popup\_window**

update() is used to change the image periodically to make the image animated, as tkinter does not support animated GIF natively.

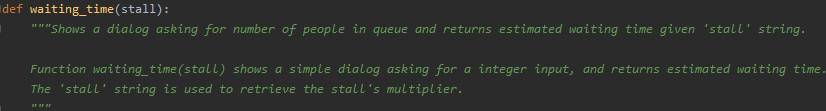
centre\_window() centres the given window by doing some calculations of the screen geometry and the window geometry



create\_elements() creates all the required GUI elements for the window

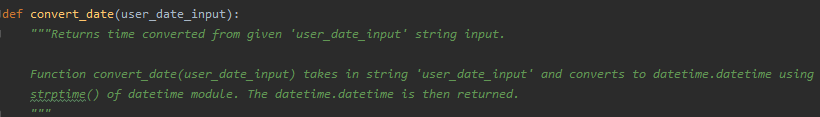


waiting\_time() lets user input number of people and get waiting time

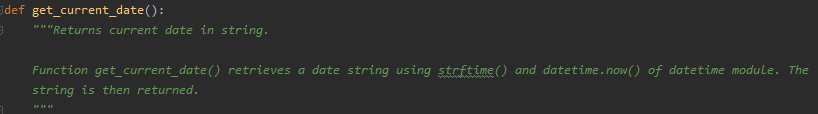


**day\_time**

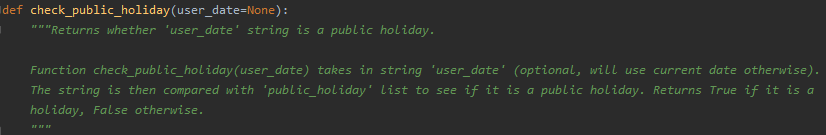
convert\_date() to convert date to string (same function exists for time)



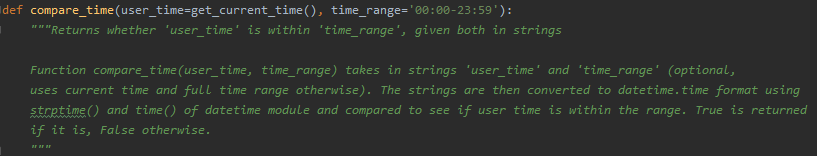
get\_current\_date() to return string of date (same functions exist for day and time)



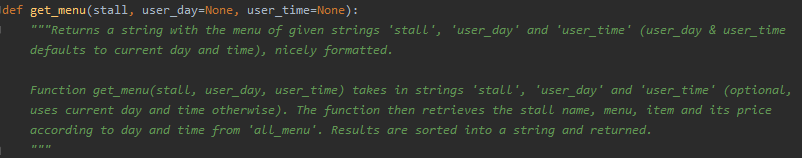
check\_public\_holiday() to check if date is public holiday



compare\_time() to check if time is within start and end time

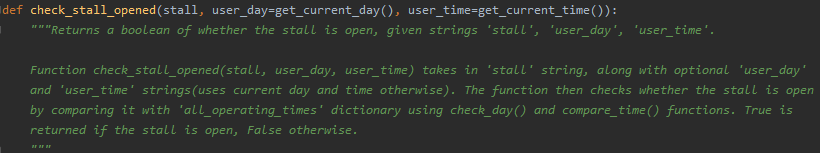


get\_menu() to get the menu of a specified stall

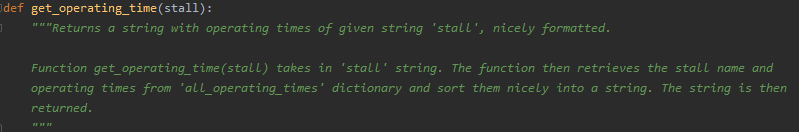


**operating\_times**

check\_stall\_opened() to check if given stall is opened

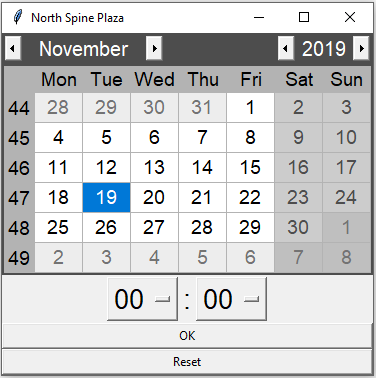


get\_operating\_time() to get operating hours of a stall



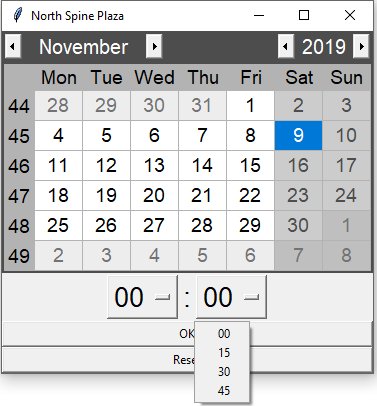
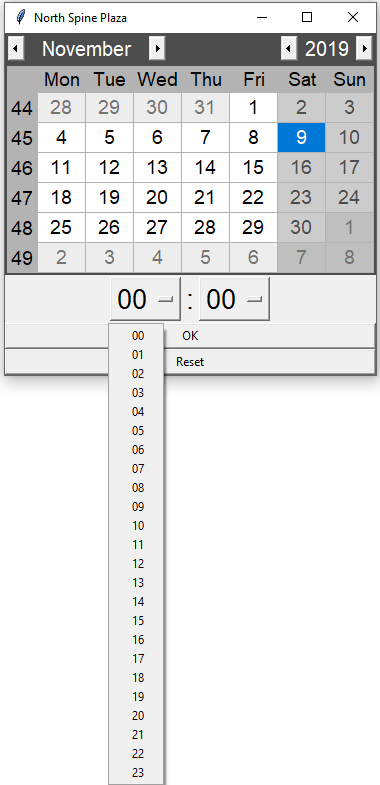
**5) Error handling test cases**

* Using tkcalendar module reduces the chances of bad input as we limited the number of choices that the user can make. The user can only select a valid date from the given calendar.



(Fig 13: tkcalendar widget)

* We used option menu for time selection in tkinter, which limits their choices to valid inputs.



(Fig 14: Choosing specific time)

If users make a wrong input, he can use the reset button to reset the selected date/time.

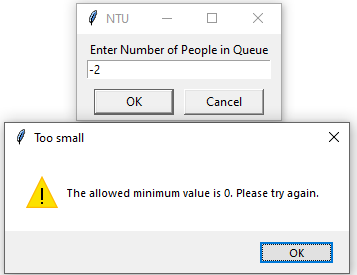
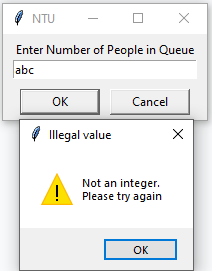


(Fig 15: Reset button)

* As mentioned above, if the user selects a stall that is closed, the program will indicate that the stall is closed.
* The second window is not resizable, so the GUI elements will not look weird.
* The close button on the top right is mapped with the same functionality as the back button.



For the waiting time estimator, users are expected to input a non-negative number (since the number of people cannot be negative). If the input is invalid (negative integer/a string), the program will show the type of error and ask for the correct input again. This process is looped until a valid input is received.



(Fig 16: Non-negative value for waiting time estimator)

**6) Individual reflection**

Venkat’s reflection

The most difficult thing that I encountered while working on the project was on how to use tkinter. tkinter was not taught in the course and even though it is like python, it has different parameters and function from python. It took some time to familiarise myself with tkinter. Furthermore, in my team I was the only one with no programming background before university. As a result, my team took a longer time to complete the program. One thing I learned while I was doing this project was that, when we are doing a program, it is necessary to test-run every line and function to see if the program works. This also makes it easier to debug the program. One thing that I feel we could improve was that, we could include more functions such as random food generator for people who could not choose a stall to eat at.

Jun Hong’s reflection

One difficulty I faced in this project is letting my teammates understand my code. Thus, I tried my best to write comments and use docstring. I used keyword instead of positional arguments to for easier reading of function and separated functions to different modules for better organisation. I also learnt that simple and readable code is better.

Last difficulty I faced is deciding which file format to use. I tried CSV at first, but it looked ugly. I also tried Pickle, but it was difficult to input new data inside because our program lack input features. I settled with json, which is more readable and easier to read in.

One thing we can improve on is to use Class. It allows private namespace for all private functions and makes the code easier to organise. I could also have imported the modules with prefixes, which makes it easier to know which module’s function I am using. Other things we could have explored is a location picker, or to pull data off the website directly.

Chin Yi’s reflection

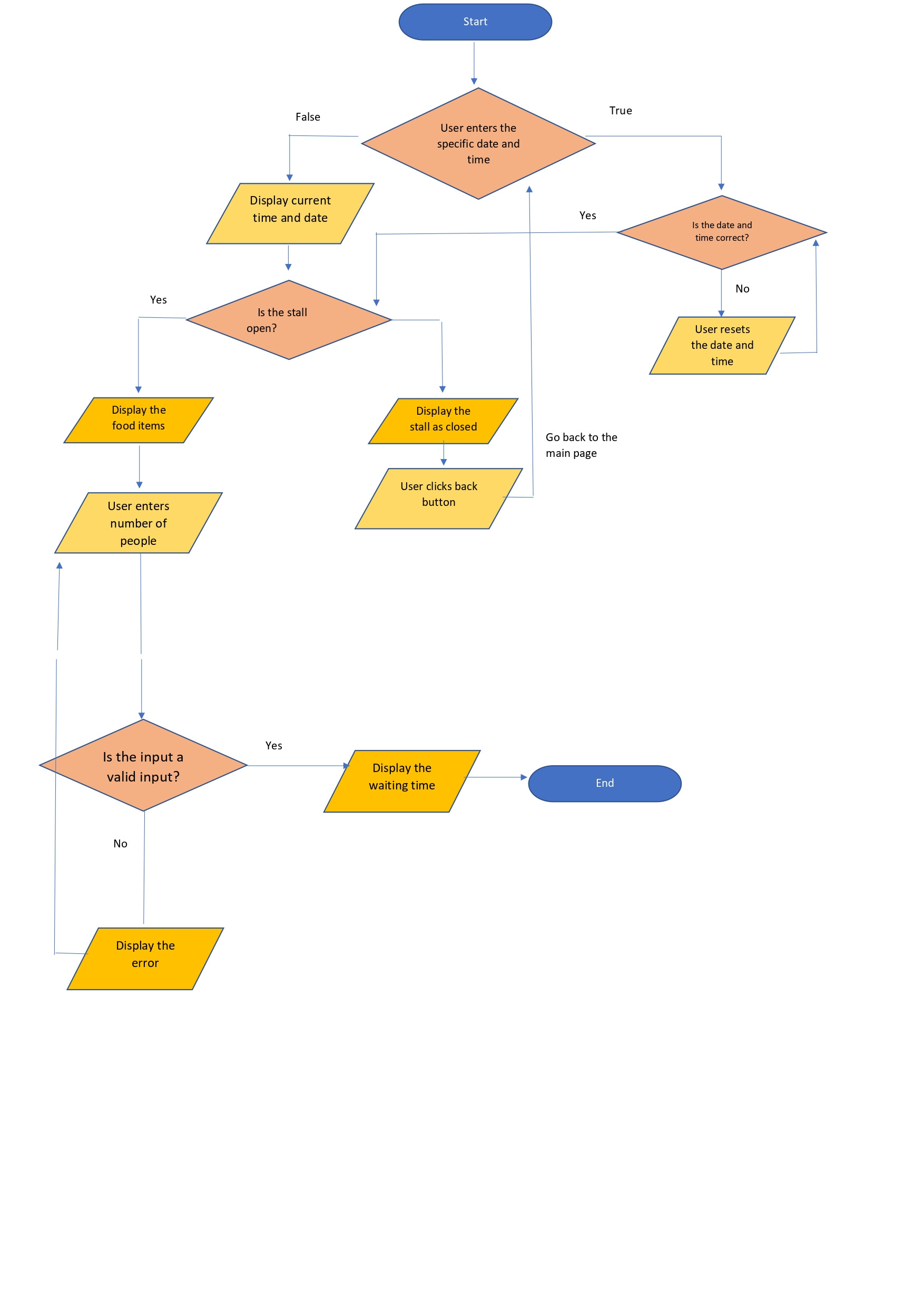
After my first time working in a team to code a program, I learnt that comments are necessary since my teammates are working on different modules that will combine to run the programme. Comments standardise the module names and help us understand the functions.

Moreover, GUI (tkinter) is a new thing so I had to google and look through the documentations and sample codes to learn to include functions from creating elements (e.g. pack, place, grid) to utilising classes for different frames or toplevel to create the windows (transitions).

One limitation I faced is that to create elements (labels, buttons, canvas etc), I must insert codes at various places, for instance under create\_element, main\_window then window\_2 function. Hence it may be troublesome when adding new elements.

One difficulty for me is that I find the parameters hard to understand initially (e.g. Parent widget, master = None etc), but after several trial and error I was able to see how different parameters bring about different effects.

**Flowchart**



Modules used

tkcalendar <https://pypi.org/project/tkcalendar/>

Image Sources

Hour Glass GIF <https://media.giphy.com/media/xFmuT64Jto3mRO4w3G/giphy.gif>

Back Button Image <https://www.freeiconspng.com/uploads/undo-back-return-button-png-3.png>

NTU Stall images <https://www.ntu.edu.sg/has/FnB/Pages/NorthSpine.aspx>

NTU Map <https://maps.ntu.edu.sg/maps>

NTU Logo <http://www.ieeenmdc.org/files/2017/02/NTU-Logo-full-colour.png>

The Hive <https://static.businessinsider.sg/2018/10/NTU-the-hive.jpg>

Other images are self-designed

Other Sources

Stall Data <https://www.ntu.edu.sg/has/FnB/Pages/NorthSpine.aspx>

Public Holiday [https://rjchow.github.io/singapore\_public\_holidays/api/\*\*<year>\*\*/data.json](https://rjchow.github.io/singapore_public_holidays/api/**%3Cyear%3E**/data.json)

**Word count:**

**Other sections: 699**

**Reflections: Venkat (152), JH (173), CY (165)**

**Contribution of each group member is on the content page.**