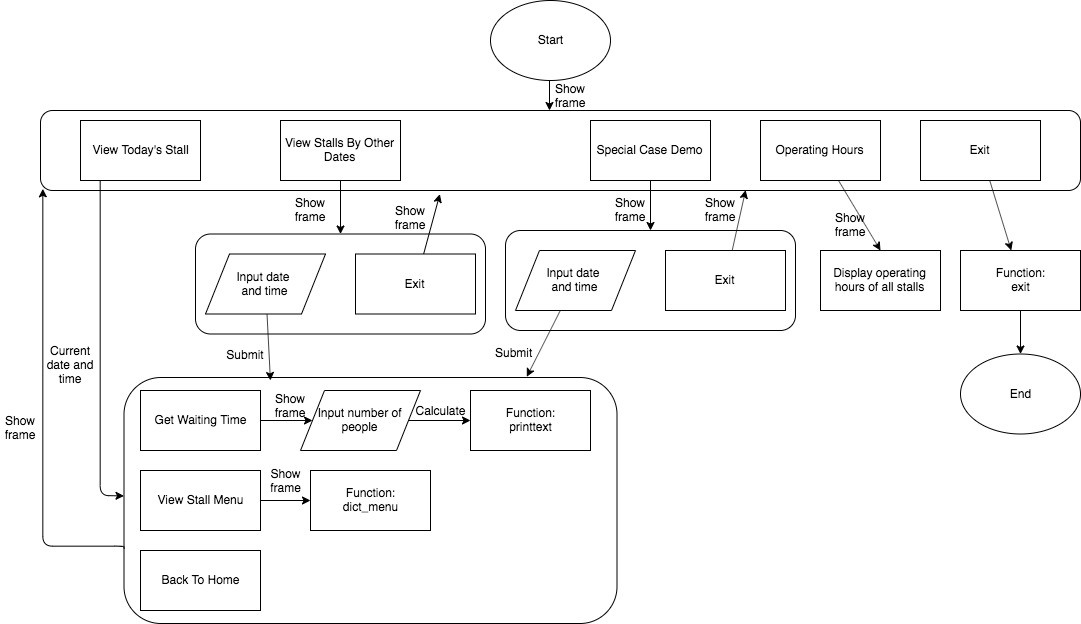
**Introduction**

This report would be an overview of how our group came up with the Canteen Application. Some things that will be included are the thought process and challenges faced.

**Top level flow chart**

****

**User defined functions**

get\_menu function: Used to retrieve the menu of each stall from the database and display it on the program.

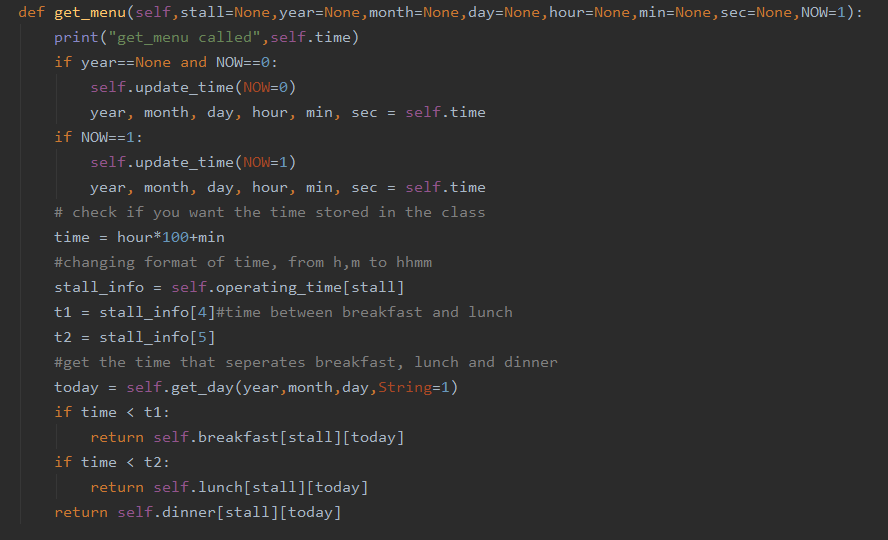
****

Figure 1: get\_menu function

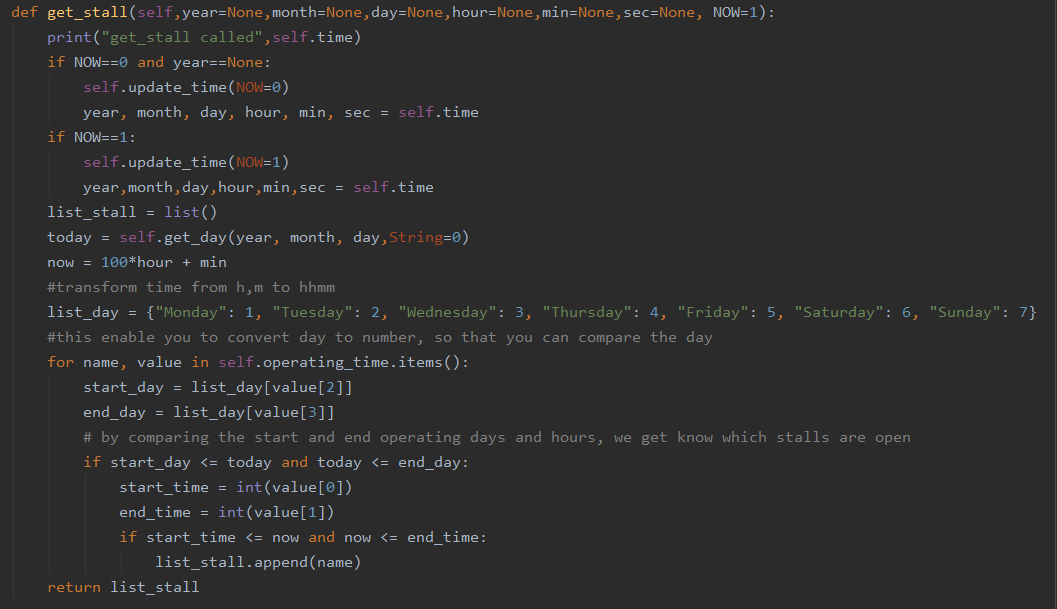
get\_stall function: Used to retrieve the stall information and display it on the program.

Figure 2: get\_stall function

input\_time function: Allows user to set the time and view the relevant stalls and menu available.

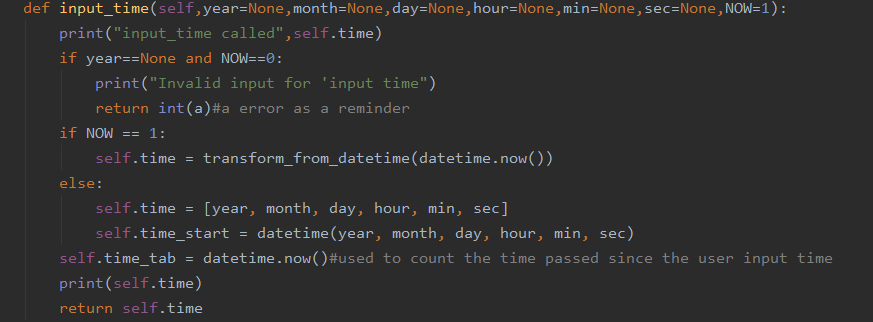


Figure 3: input\_time function

**Exception handling**

We have two instances where user inputs are required, one for calculating waiting time and one for setting the date and time to view. Hence, there are two cases of exception handling.

In order to calculate waiting hours, a user input on the number of people in the queue is required. If the user does not input a number (i.e a symbol or an alphabet), the program will have a pop up stating “ invalid input, please enter a valid number”. If the user inputs a zero or a negative number, the program will have a pop up stating “invalid input, please enter a number that is at least 0”. Hence, only when the user inputs a number more than zero will the waiting time be generated.

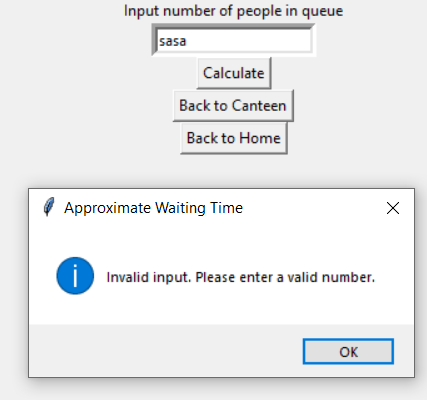


Figure 4: Input a non-number in waiting time

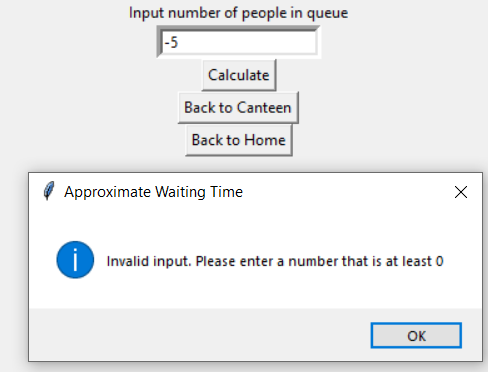


Figure 5: Input a negative number in waiting time

Users can check what stalls are available by inputting the date and time they would like to view. When the user inputs a symbol, alphabet or a non-existent date (eg. 40 in month input), the function will check if the input corresponds to actual year, month, day, hour, minutes and seconds. If it comes up with an error in any of the columns, the user’s input will be cleared, allowing them to try again.

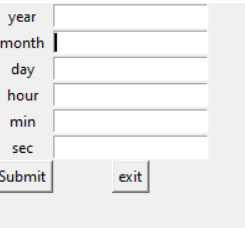


Figure 6: time resets with wrong input

**Ting Qi’s reflections**

I did the database for the food menu and had difficulties in formatting the text file into a dictionary when reading it in Python. To tackle this, I imported the text file into the python file and used ast(a python library) to convert it to a dictionary. Realising this method is prone to errors, I had to check each line of the database after writing it. Also, as using ast evaluation removes decimals of food prices, I used Python’s locale library to convert it into money formatting.

For the operating hours, I could not use the same method ast.evaluation() as the ast function does not allow a list in a dictionary. Hence, I wrote the operating hours as a normal text file and used functions to change the text into a list in a dictionary. I ran into some problems while doing so, such as having to use strip function in python to remove new line character in python and having to figure out how to create a dictionary in a list using for loops to loop through the list and using key value pairs to initiate them.

Overall, I have learnt a lot on file handling, file management, database debugging and string manipulation from just creating and using the database. I could have improved in a few areas such as helping to pull the data from the python dictionaries when time function is used, create more functions for the pulling of data from the databases and comment on the functions used with more precision.

**Yu Peng’s reflections**

I did the functions which enables user time input, read user time and to get menu based on time.

Firstly, I had problems with the program opening the file every time it calls the function for stalls and menus. I chose to read the data in a class instead so that the file is only read once when the program is run. This allows greater program efficiency when it runs.

I also had problems getting the app to return to displaying the stalls available at the current date and time after it has been filtered by the user input time. I was able to solve this through the creation of a refresh function, allowing users to refresh the page so that it will return to the current date and time after the user is done with the information for the user inputted date and time inputs. However, I then ran into the problem of having the widgets repeatedly flash when refreshing the store menu. To fix it, I placed the relevant buttons before forgetting the unwanted.

Lastly, I also had some problems with the checking of validity of special dates like 30th February on non leap years. To solve this, I used the time library to construct datetime object which can spot the special dates and shows an error when an invalid date is inputted.

Overall, I learnt a lot about datetime objects, gui handling and dealing with user input errors. I could have improved on the commenting of the functions for ease of reading and writing more efficient code in the future.

**Jolene’s reflections**

The main GUI and several functions such as going back and forth between pages and calculating waiting time were done by me.

Firstly, I had problems opening a new page and going back and forth the different pages of the GUI. Initially, the windows would stack up every time a new page is open and it was only possible to return to the previous page opened. To solve this, I thought of using different frames in a single class and putting them in a list. I then created a function to call the different frames and raising them to the top to be displayed. The different frames would fit into the geometry of the class and hence prevent stacking. Users could also switch between pages easily as I created a function to tie different buttons to open specific frames.

I also had problems with fitting photos into the application. The photos my group found were of various dimensions different from the size of our application. I solved it using the resize function which allows me to crop the image according to the dimensions I desire.

Finally, I had problems with displaying the output of the waiting time on the GUI instead of on the backend code. This was solved by learning to use the message box feature which displays the output to the user in a clear and concise manner.

In conclusion, I learnt a lot about using Tkinter such as how to raise and destroy different frames as well as tying different functions to specific buttons. Although there were many online tutorials for this, the challenge was how to apply what you have learnt such that it is relevant to your project and combining the different features all in one main code. However, I felt I could have added a back function in every page that allows the user to go back to the previous page. Also, I could have figured out a way to automate displaying of the stall operating hours instead of hardcoding it.