# 

# 

# **CZ1003 Mini Project**

Real Time Canteen Information System

****

|  |  |
| --- | --- |
| **Content** | **Page Number** |
| **Chapter 1: Introduction**  1.1 Who are we?    **Chapter 2: Program Design**  2.1 Algorithm Design  2.2 Data Storage  2.3 Welcome Page  2.4 View Menu Page  2.4.1 View Menu - “Use current date and time”  2.4.2 View Menu - “Use personalized date and time”  2.4.3 View Entire Menu  2.5.1 Additional Feature 1 - Random Food Generator (RFG)  2.5.2 Additional Feature 2 - Shortest Queue  2.5.3 Additional Feature 3 - Self-destruct windows    **Chapter 3: Reflection**  3.1 Learning Tkinter 3.2 Project Approach 3.3 Splitting the workload  3.4 Using Widgets 3.5 Additional Functions’ Accuracy  3.6 Project Contribution  **Appendix** | **3**  **4**    **5**  **6**  **7**    **8**    **10**    **11**    **12**    **13**  **14**      **15**    **16**    **17** |

# **Chapter 1: Introduction**

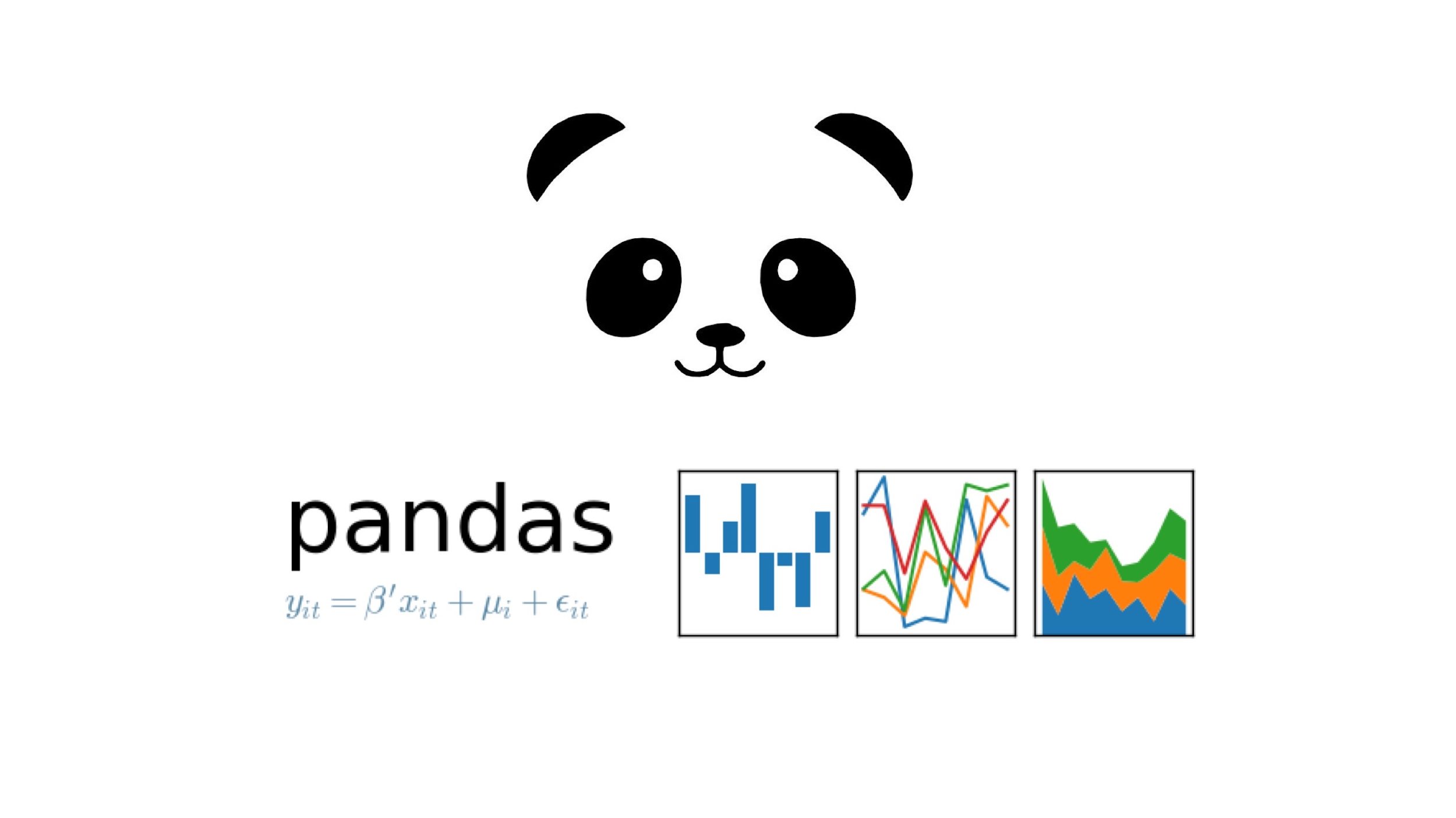
Who are we? What do we hope to achieve?

****

1.1 Who are we?

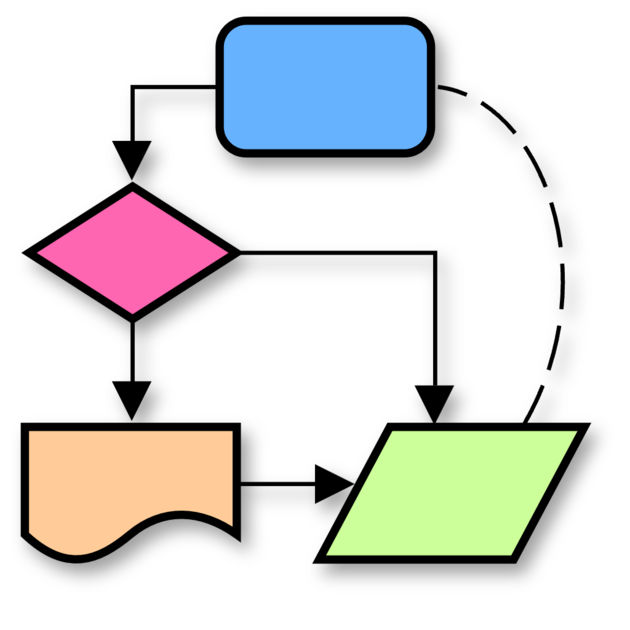
A team, consisting of Jeremy, Joshua and Ernest, designed a real-time NTU North Spine Canteen information system application.

The program was written in **Python** with **Tkinter** and **Pandas**.

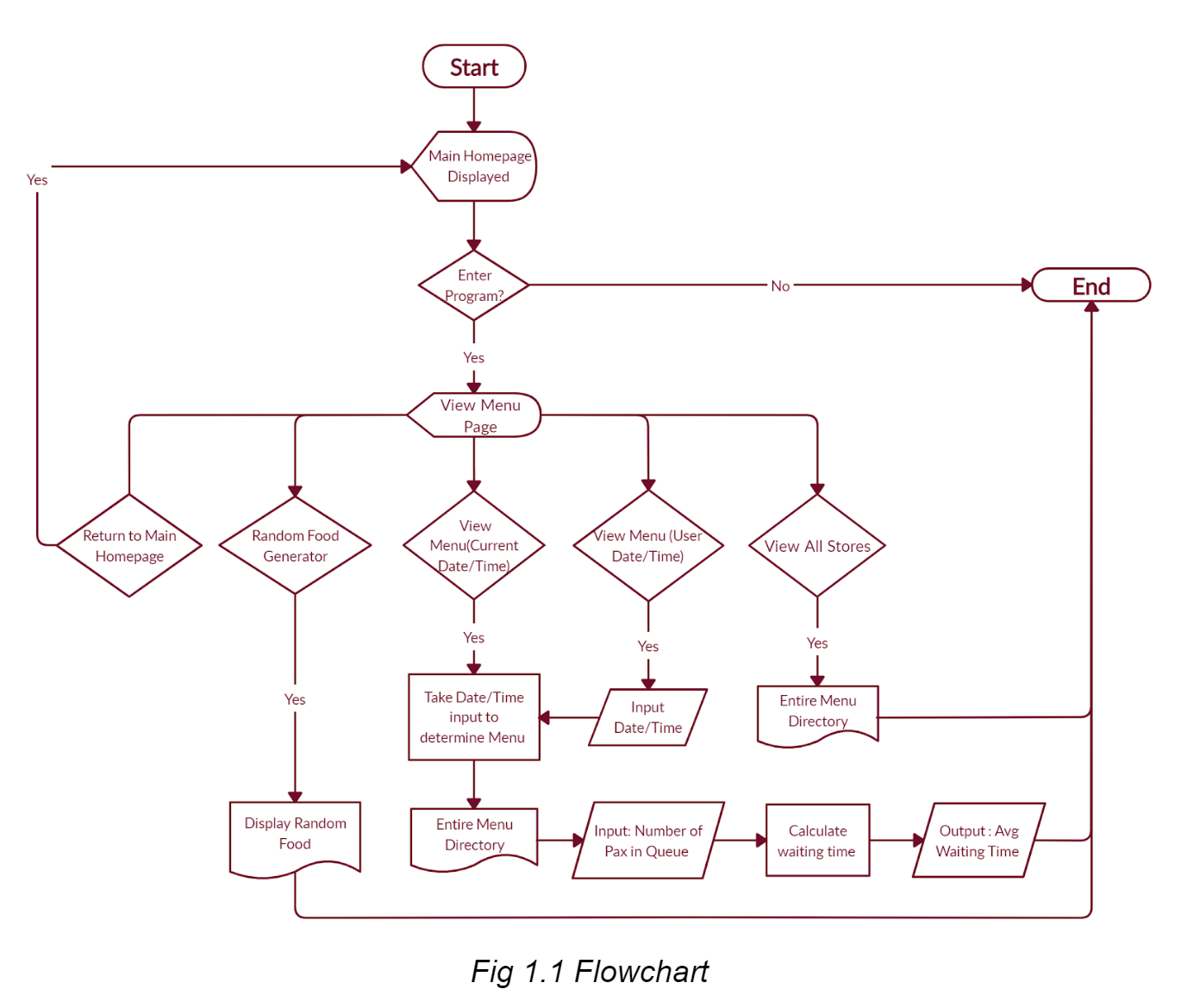
** **

**Chapter 2: Program Design**

Algorithm Design, Program Run-through, File Handling

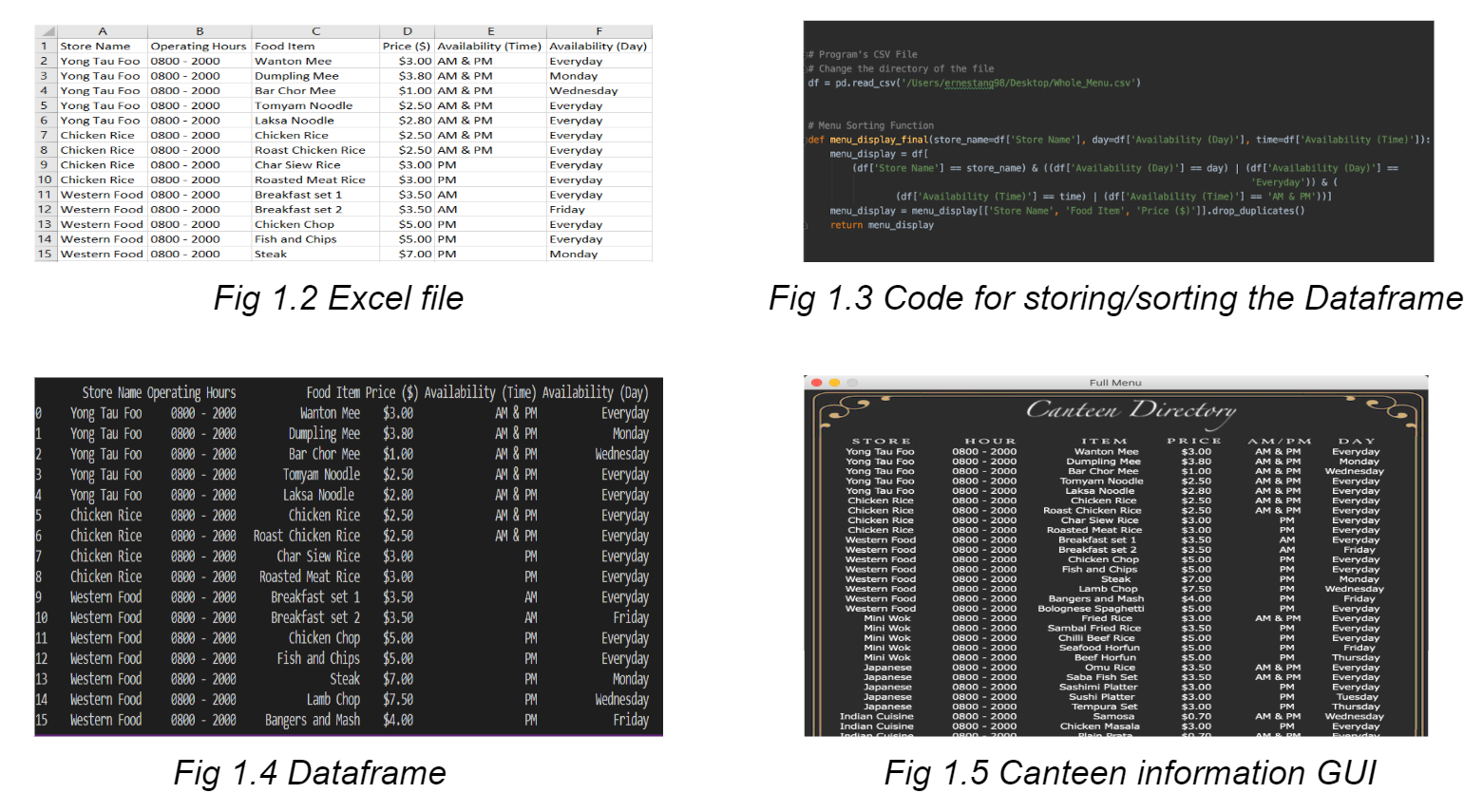


2.1 Algorithm Design



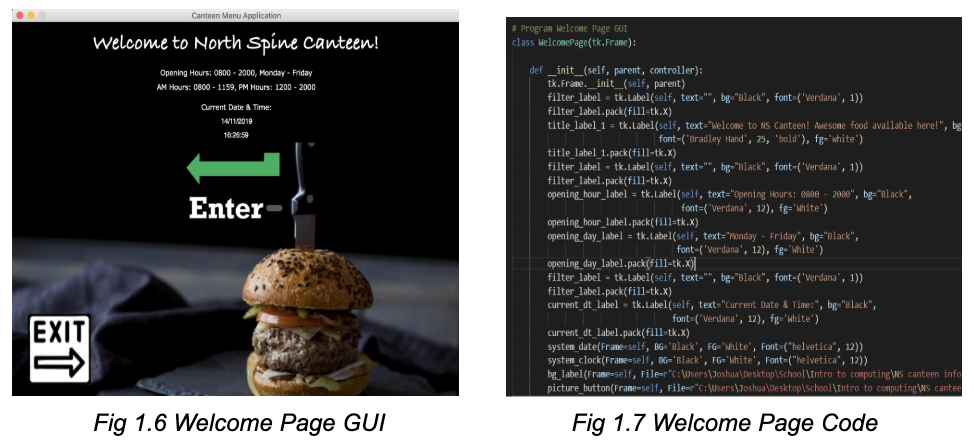
Users can view available food options based on either current system’s date/time or user-specified date/time. Essential canteen information such as price and operating hours are also displayed.

2.2 Data Storage

****

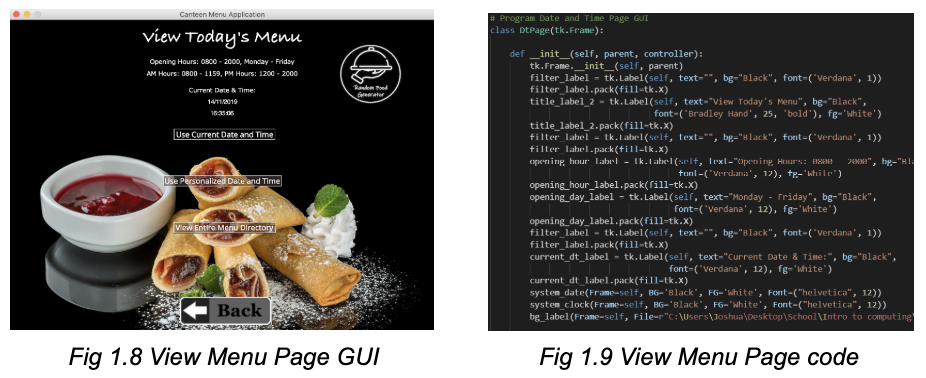
The canteen information is stored in an excel file (Fig 1.2), converted into a dataframe (Fig 1.4) using the pandas library (Fig 1.3) and displayed to users (Fig 1.5).

2.3 Welcome Page

****

Beginning with a welcome page, the canteen’s operating hours and the current system’s date and time are displayed. Users can access the View Menu page or exit the program.

2.4 View Menu Page

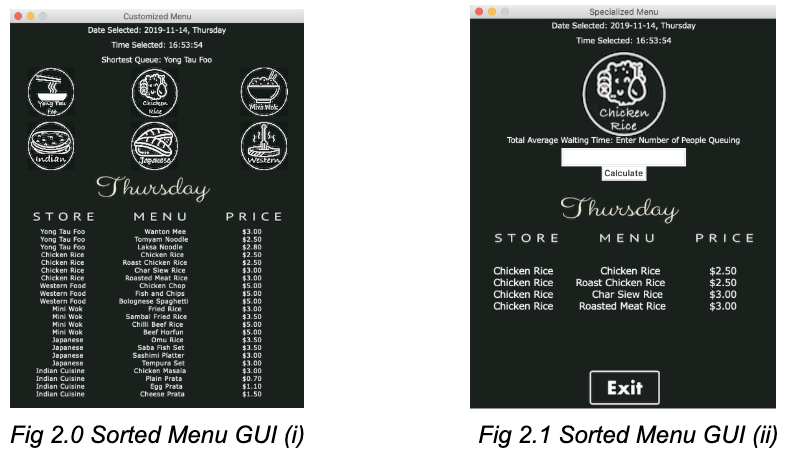


Users can either return to the welcome page or view the menu based on :

1. Current system’s date and time
2. User-defined date and time
3. View entire menu

2.4.1 View Menu - “Use current date and time”

The menu sorting function (Fig 1.3) filters the dataframe based on current date/time. A new window will display the food options available and the store with the shortest queue.



Upon clicking the store icon, the store’s menu will be displayed along with a function to calculate the average waiting time specific to that stall since different stalls have different waiting time per person.



Possible scenarios:

1. Valid input
   * Total waiting time (minutes) = number of people in the queue (user input) \* waiting time per person for that store (Fig 2.2).
2. Invalid input (non-integer, no input)
   * If ‘.isdigit()’ is false, return error message.
3. Illogical input (100000 number of people in queue)
   * Return ‘Waiting time at least an hour’ when total waiting time exceeds 60

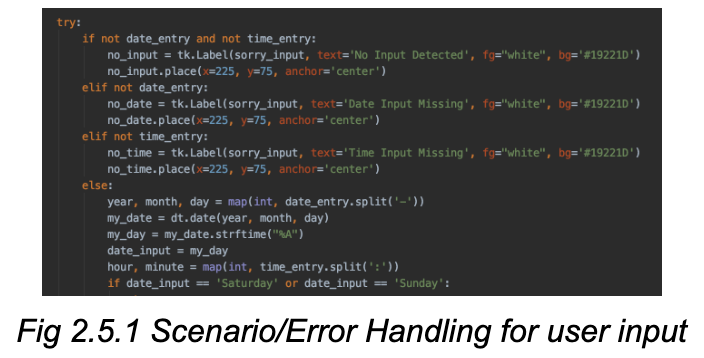
 

2.4.2 View Menu - “Use personalised date and time”

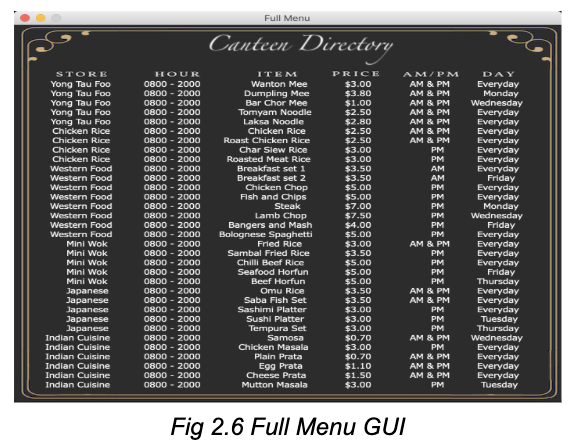
A new window opens, requesting user input for date and time.

Possible scenarios:

1. No input
   * Return ‘No input detected’
2. Only time input
   * Return ‘Date input missing’
3. Only date input
   * Return ‘Time input missing’
4. Invalid date/time input
   1. User inputs date in the wrong format
   2. User inputs time in the wrong format
   3. User input date and time in the wrong format
      * Return ‘improper format, re-enter date and/or time’ for 4a - 4c
5. Valid input
   * Returns specific menu, details seen in 2.4.1



2.4.3 View Entire Menu

**

Each column of the dataframe (Fig 2.6) is converted into a string label. These labels are then packed and displayed in a window for users to view.

2.5.1 Additional Function 1 - Random Food Generator (RFG)



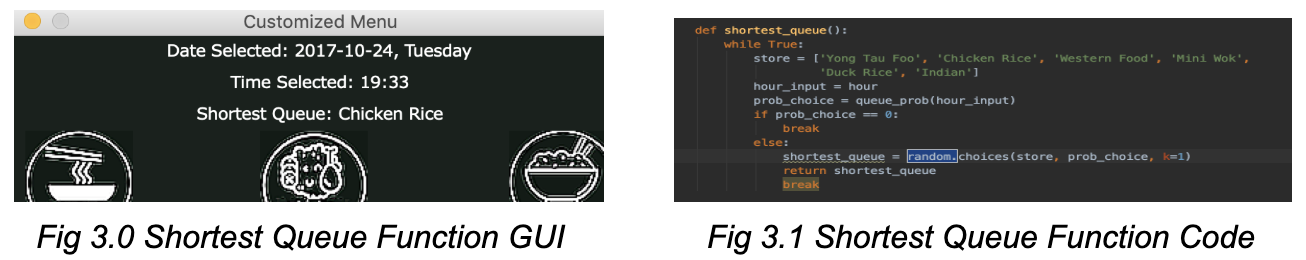
This function filters food options available based on current date/time and requests user for the number of customers to generate random food options for. Taking the user input as a parameter, the function uses *“DataFrame.sample()”* method to return a random sample of user-specified size. This information is displayed in the GUI as seen in Fig 2.

Possible scenarios:

1. Valid input
   * Return generated food options
2. Invalid input (non-integer, no input)
   * If ‘.isdigit()’ is false, returns ‘Error. Enter Digits(s) only’
3. Illogical input (more than 20)
   * Returns ‘Error. Enter up to 20 people per input’
4. Date/Time error - Outside of operating hours
   * Returns ‘Food court is closed’

2.5.2 Additional Features 2 - Shortest queue

This function filters the stores based on date/time selected and displays the store with the shortest queue based on that. Stores are given probabilities of having the shortest queue, based on the popularity of the stores at different timings. Taking the specific date/time as parameters and using *‘random.choices()’* method, the function returns that store based on the respective probability.



2.5.3 Additional Features 3 - Self-destruct windows

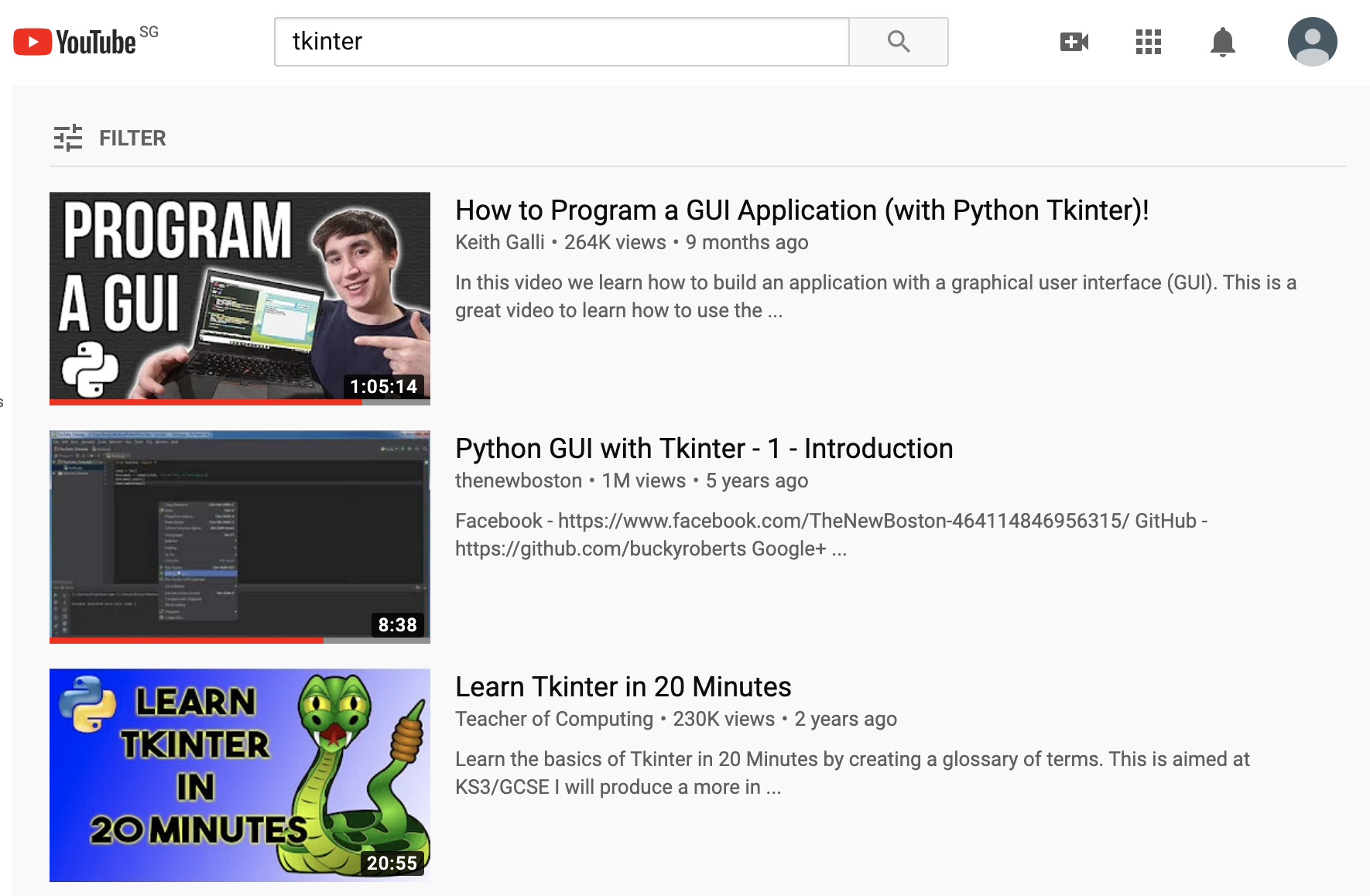
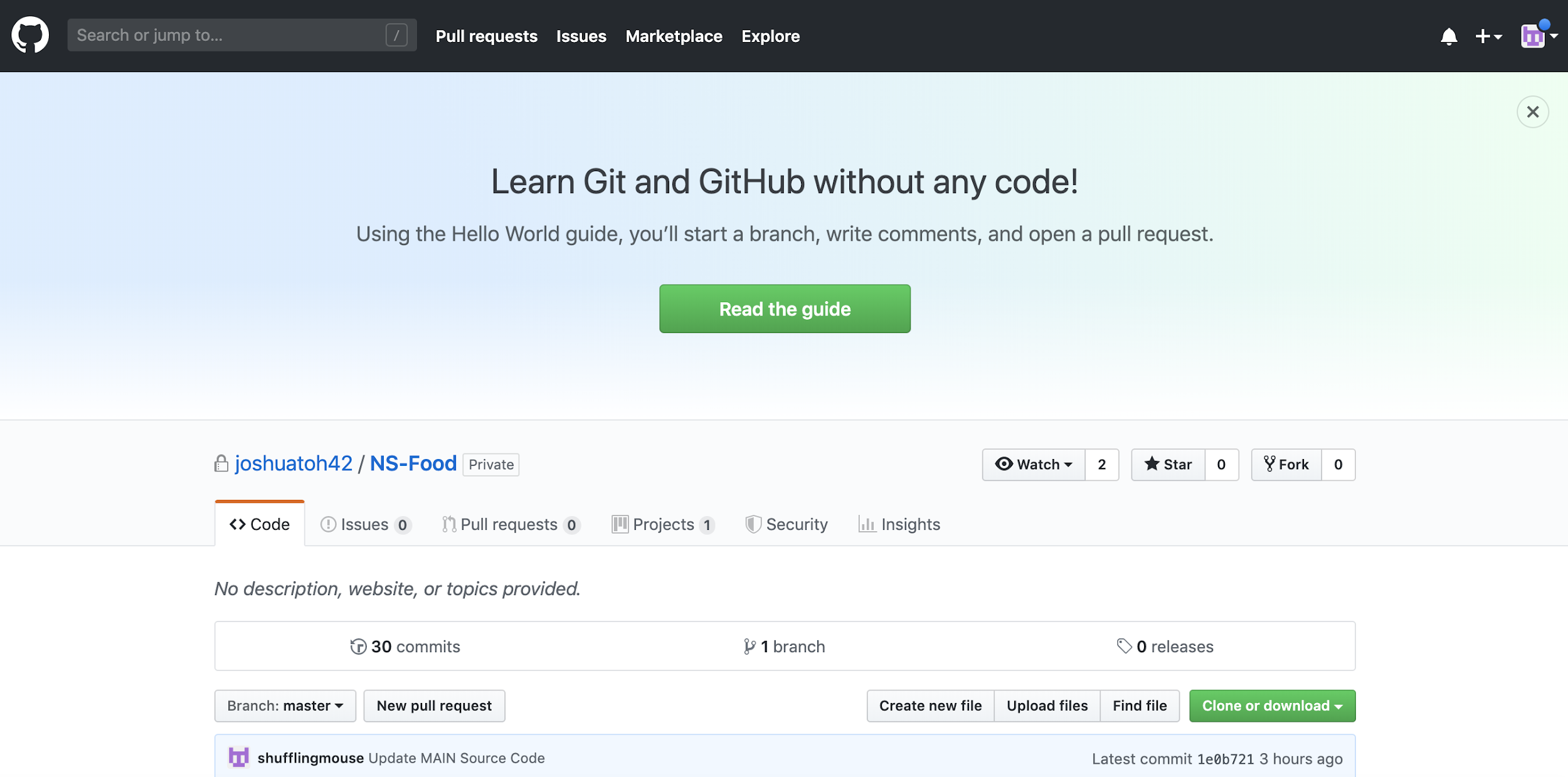
Invalid input and waiting time windows will automatically close after 3 seconds.

**Chapter 3: Reflection**

Difficulties encountered, learning points

****

3.1 Learning Tkinter

Being our first time coding a graphic user interface (GUI), we struggled. This led us on a path of self-learning, spending countless hours on YouTube tutorials and various coding forums. To smoothen this process, we created a private project on GitHub which facilitates group learning, cooperation and contribution. We were able to cover each other blindspots while learning and coding the GUI.

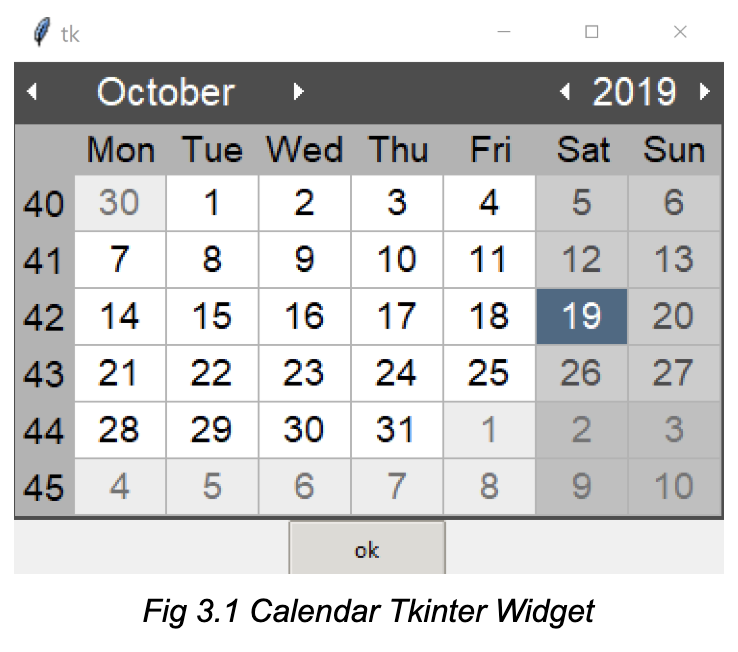
3.2 Project approach

After reading the project brief, we learnt Tkinter individually. We got overly enthusiastic trying to apply what we learn and started on the GUI immediately. Although we were able to explore and built a myriad of Tkinter functions, we lacked a concrete plan which only slows our progress of building the program. Subsequently, we first mapped out the necessary details such as flowcharts and required functions. This resulted in us staying focused and clear of the task at hand. To keep everyone updated, we also held weekly meetings which prove critical to the project.

3.3 Splitting the workload

We segmented the project into 2 parts: the front-end GUI and back-end code. Our plan was to finish them individually and combine them into one massive source code. Initially, integration was easy as the components were relatively small. As our code got larger, combining codes became problematic as our syntaxes are different. For example, Joshua’s GUI for the sorted menu was done through ‘Pandastable’ library. However, Ernest’s code requires the code to be in a string format and the function had to be converted into other formats. Moving forward, communication is essential to avoid building incompatible codes.

3.4 Using Widgets



When requesting for user inputs, providing options using Tkinter’s widgets instead of open-ended responses from users might prove to be successful in reducing input errors. For instance, the ‘Calendar’ widget may be used to request for date input. This reduces date input errors and hence, reduces the complexity of file and error handling as the user cannot deviate from the choices that are presented by the program.

3.5 Additional Functions’ Accuracy

For the shortest queue generator, the information we coded was based on our experiences as patrons of the North Spine Canteen. We could have further improved the accuracy of the function by gathering factual or even real-time data on the average number of people queuing for a particular store at a particular time, and on a particular day

3.6 Project Contribution

|  |  |
| --- | --- |
| Name | Contribution |
| Ernest | Chapter 1.1 — 2.3 & 3.1  Create GUI in Tkinter  Create user input  Create welcome page  Create visuals for menu |
| Joshua | Chapter 2.4 — 2.4.3 & 3.2 — 3.3  Create pandas function  Create date time page  Create visuals for buttons  Error handling |
| Jeremy | Chapter 2.5.1 — 2.5.3 & 3.4 — 3.5  Create clock function  Create view menu page  Create visuals for Home page  Create calculate waiting time function  Create additional functions  Create GIF |

**Appendix:**

Report’s Image Source

Page 1:

1. Cafeteria Clipart, retrieved from <https://www.pngrepo.com/svg/132681/cafe>

Page 3:

1. People Clipart, retrieved from <https://icon-library.net/icon/who-we-are-icon-28.html>

Page 4:

1. Python Logo, retrieved from [https://commons.wikimedia.org/wiki/File:Python-logo-  
   Notext.svg](https://commons.wikimedia.org/wiki/File:Python-logo-notext.svg)
2. Pandas Logo, retrieved from <https://encrypted-tbn0.gstatic.com/images?q=tbn:AN>  
   d9GcSQTysT6IrVEe35oHNnm5zckg6S5wU4oWt4VYnxOEU2rNBKx7c\_og&s

Page 5:

1. Flowchart Clipart, retrieved from [https://icon-library.net/icon/flow-diagram-icon-  
   4.html](https://icon-library.net/icon/flow-diagram-icon-4.html)

Page 13:

1. Thinking Clipart, retrieved from [https://thenounproject.com/term/self-reflection  
   /114599/](https://thenounproject.com/term/self-reflection/114599/)