Ver 1.0

Task 0 : Explain what you are doing/ going to accomplish

Get my python server running, including constructing my class with my test data

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*

N/A

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Will have a class that contains my menu and the variables (canteen\_food). Will help with displaying my info to the user when linking python to HTML.

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

N/A

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

N/A

Task 5: Identify any constants or existing data if required

Test data:

* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12
* Ham and Cheese Sandwiches - 4 in stock

Task 6: Identify indexed data structures

Canteen\_test , will contain my test data which is nessecary for later when I put this data into my website.

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

N/A

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

The initialise function, gives each food item a name, stock, price and description

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

From bottle import run, route, view, get, post, request, static\_file

Form itertools import count

Class canteen\_food

Set \_ids to count(0)

Define \_\_initialise\_\_(self, name, image, stock, description)

Set self.id to next(self.\_ids)

Set self.food\_name to name

Set self.food\_image to image

Set self.food\_stock to stock

Set self.food\_price to price

Set self.food\_description to description

Set Canteen\_test to [

Canteen\_food(“Sushi Roll Pack”, “image”, “5”, “price” “description”)

Canteen\_food(“Hot Dog and chips”, “image”, “12”, “price” “description”)

Canteen\_food(“Ham and Cheese Sandwich”, “image”, “4”, “price” “description”)

]

Run(set host to “0.0.0.0”, set port to 8080, set reloader to true, set debug to true)

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

N/A

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

If there is no error messages then I can expect the code to be working, although I will have no way of testing this till version 2 when I create my index page.

Task 12: Refine the plan

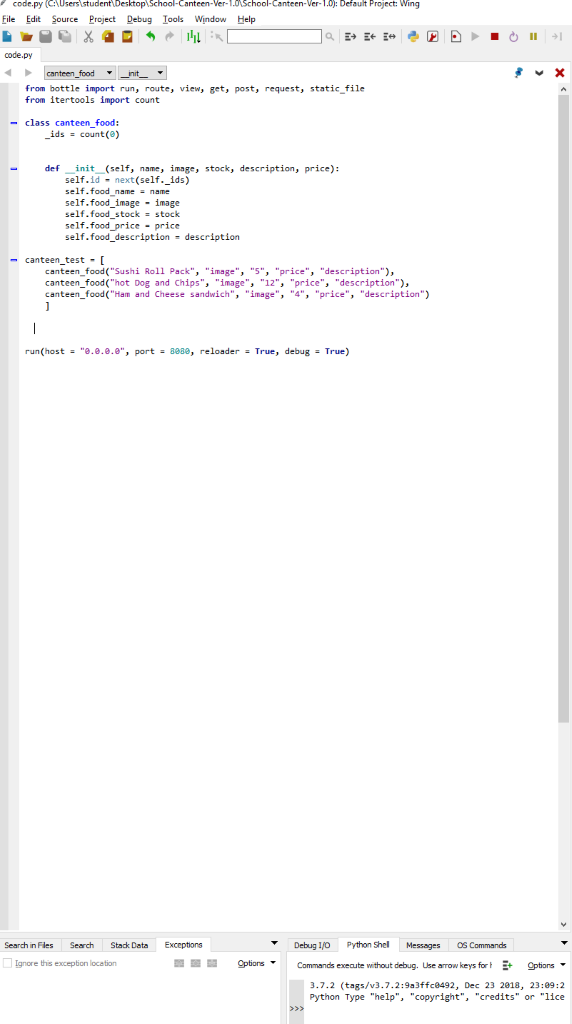
*Note any modifications here when iterating through the development cycles.*

*Needed capital T for true on the last line*Run(set host to “0.0.0.0”, set port to 8080, set reloader to true, set debug to true). Also when giving arguments to my class I forgot to include price so I was using but not giving an argument which gave an error.

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

The below picture shows my program running without any errors. I will know just how well this works in my next version when I add my index page.



Task 14 : Evaluation

*How did your version turn out*

Good, will know the full extent of how well it works when I add my index page in the next version.

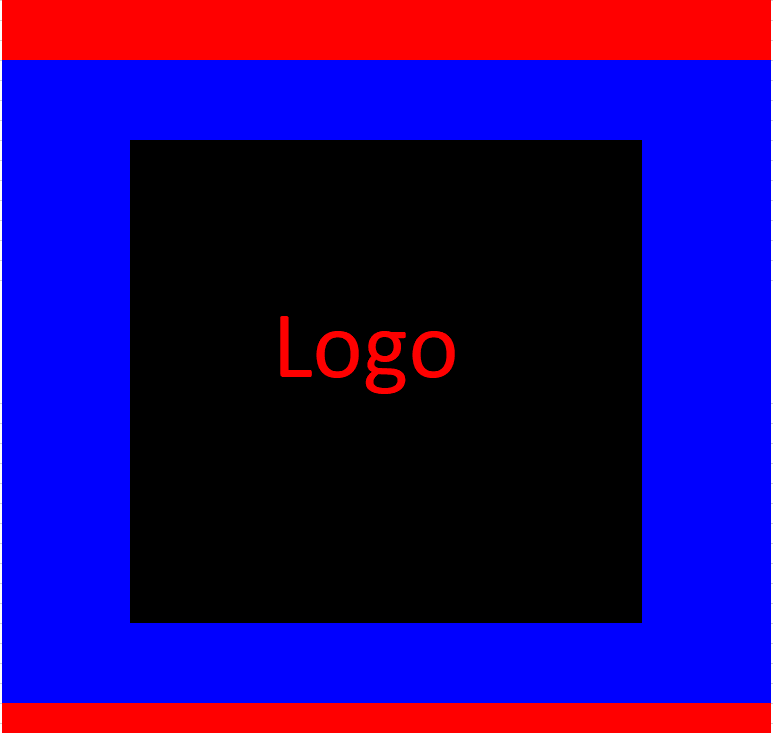
**VER 2.0**

Task 0 : Explain what you are doing/ going to accomplish

To display an index page to the user, with the appropriate colours and nav bar, also a footer.

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*



This is what my index page will look like, the black representing the school logo.

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Canteen\_food, contains the \_\_init\_\_ function which creates my objects

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

My index page, with a nav bar, footer, logo, title and the right colours.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

They will be able to click the links on the nav bar but they will not lead anywhere at this stage.

Task 5: Identify any constants or existing data if required

* Test data:
* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12

Ham and Cheese Sandwiches - 4

Task 6: Identify indexed data structures

N/A

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

N/A

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

At route (“/”)

At view(“index”)

Define function index():

Pass

For index page:

% include shared / header.html

% include shares / navbar.html

% include shared / footer.html

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

The website should be easy to navigate around and have clear option paths.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

Running my python code and loading my webpage, because there is no user usability if the webpage loads with the nav bar, logo and footer it will be a success.

Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Forgot to add the function that allows pictures

To display images:

At route(‘/picture/<filename>’)

Define saved\_pics(filename):

Return static\_file(filename, root = ‘./images’)

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

Task 14 : Evaluation

*How did your version turn out*

Very good, once I figured out why the picture didn’t load and fiddled with the red colour a little bit it came together quite nicely.

**Ver 3.0**

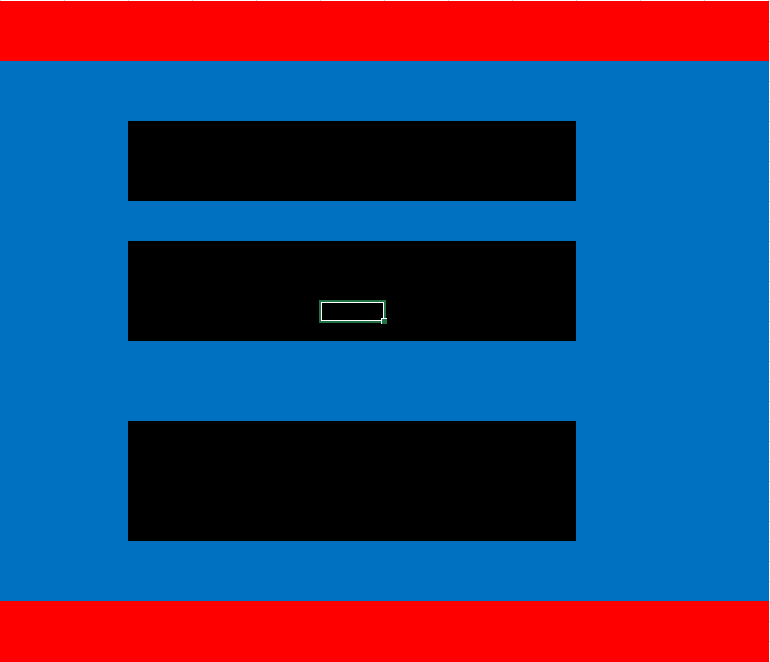
Task 0 : Explain what you are doing/ going to accomplish

Create a terms and conditions page, as well as a menu page.

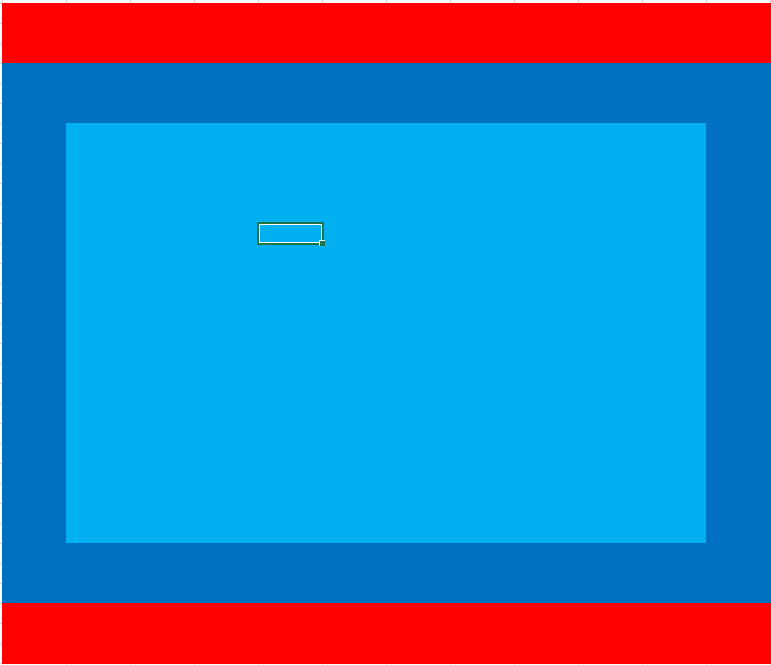
Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*

Menu page



Terms and conditions page, light blue represents where the writing will go.



Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Canteen\_food, contains the \_\_init\_\_ function which creates my objects

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

The index page which was created in the last version. The links on the navbar will have to be easily accessible and easily seen. The links will go to pages which are displayed clearly.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

They will be able to navigate around my website using the nav bar.

Task 5: Identify any constants or existing data if required

* Test data:
* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12

Ham and Cheese Sandwiches - 4

Task 6: Identify indexed data structures

Canteen\_test, contains all my test data

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

N/A

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

Has the initialize function, which assigns each food item name, price, description and stock

The function for each of my web pages which means I can access them off my website as they are on the server, also means I can do any calculation necessary for that webpage to have.

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

At route (‘/food’)

At view (“food”)

Define menu\_page():

Set data to dict ( set food\_list to canteen\_test)

Return data

At route (“/tanc”)

At view (“tanc”)

Define terms\_page():

pass

python in my HTML, this will be on all webpages

%include('shared/header.html')

%include('shared/navbar.html')

%include('shared/footer.html')

also in my “food” page I have a for and an if loop.

%for food in food\_list

%if food\_stock > 0:

%else:

%end

%end

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

Legal – terms and conditions need to be easily read, have used non complicated language.

Functionality – all links need to go to the right webpage – have named each link accordingly

Usability – website needs to be easily navigated around, this includes errors and success pages – will add success pages in next version, have not found any errors yet.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

The user should only be able to be able to go to different pages in my website so all links needs to link to the right place. There is no other user functionality at this stage.

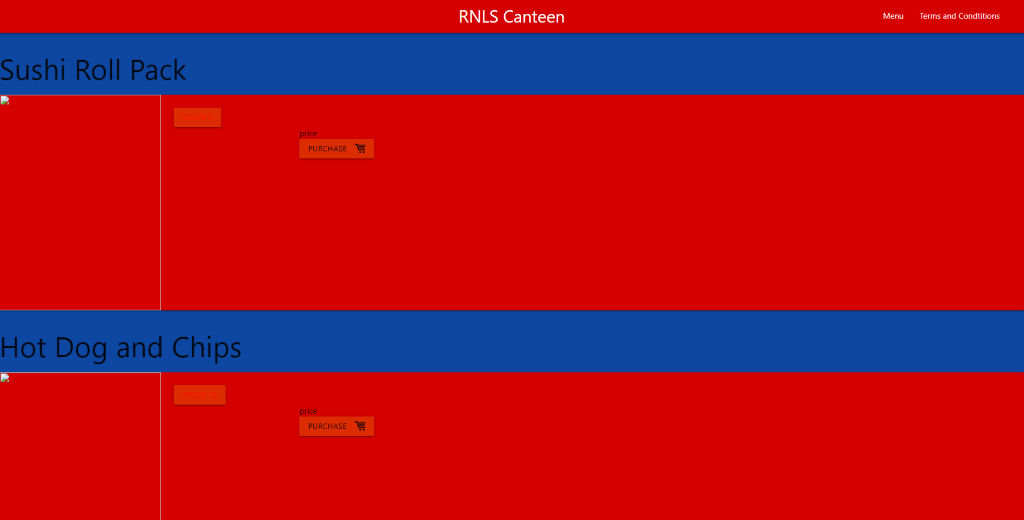
Task 12: Refine the plan

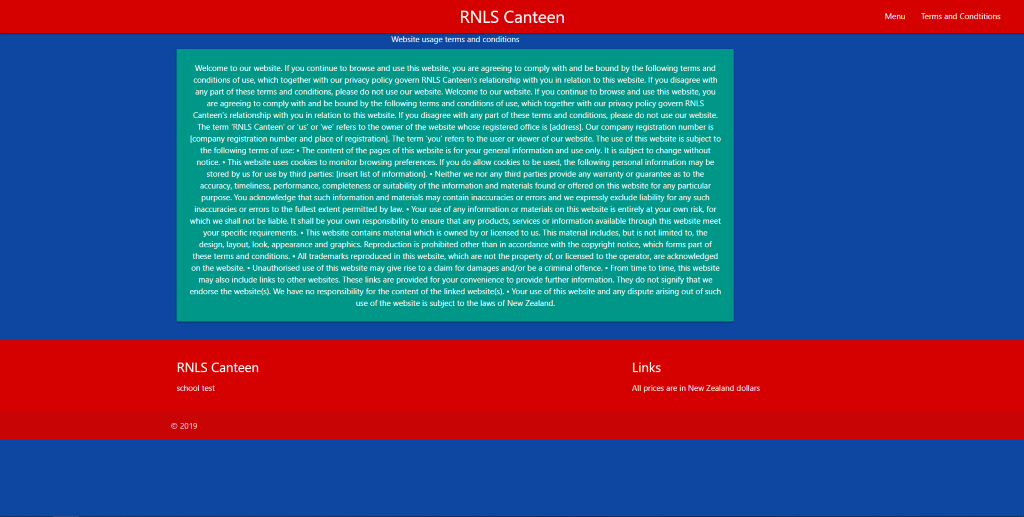
*Note any modifications here when iterating through the development cycles.*

Had to add a container around my card with the terms and conditions to centre it. But didn’t work very well.

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*





Task 14 : Evaluation

*How did your version turn out*

Good, everything worked well.

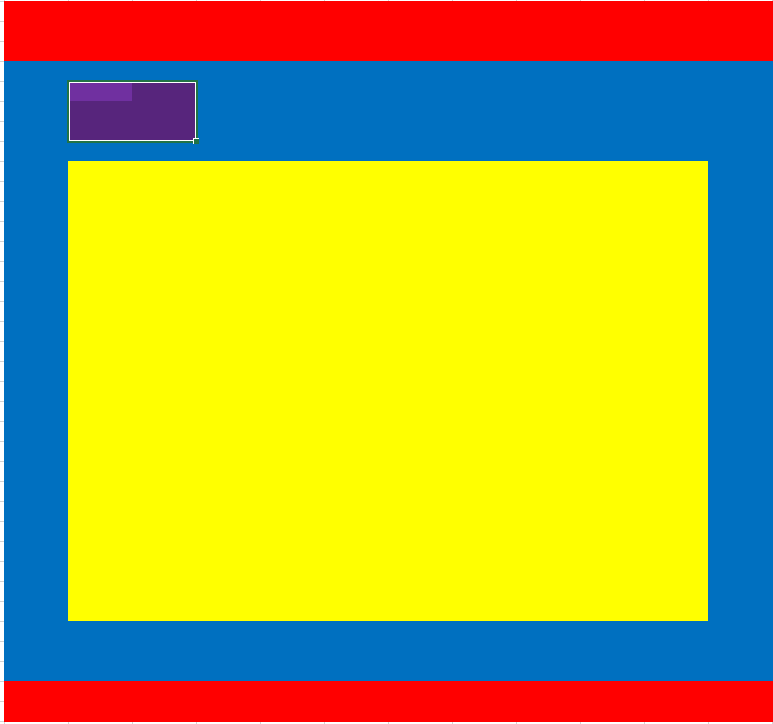
Ver 4.0 (ver 1.4 in git hub)

Task 0 : Explain what you are doing/ going to accomplish

In this version I will be adding the pictures, description and stock to my menu. Along with a buy button that links to a success page if food is bought successfully.

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*



Purple represents the back button and yellow represents the picture for the purchase success page.

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Canteen\_food, contains the \_\_init\_\_ function which creates my objects

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

The success page will be crucial as they need to know what is happening. Also the menu will need to be displayed with the correct pictures so the user knows what they are purchasing.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

They will be able to order food, this will minus one off the stock each time, this will line to my order success page.

Task 5: Identify any constants or existing data if required

Test data

* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12

Ham and Cheese Sandwiches - 4

Task 6: Identify indexed data structures

Canteen\_test – conatins my test data

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

When the user purchases food the program will have to minus one off the stock of the particular food item they bought

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

The only function I will add in this version if the success page function, this will match the food id and minus one stock off it each time it is bought.

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

At route(“/success/<food\_id>”)

At view(“success”)

Def success\_page(food\_id):

Set food\_id to the integer of (food\_id)

Set found\_food to none

For food in canteen\_test DO

If food.id is equal to food\_id DO

Set found\_food to food

Set data to dictionary(set food to found\_food)

Set Found\_food.food\_stock to minus 1

Return data

In success page

%include('shared/header.html')

%include('shared/navbar.html')

%include('shared/footer.html')

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

The user will need to easily navigate from the purchase success page back to the menu page.  The success pages will tell the user if their purchase has been a success. Have also added some comments in my python for readability.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

Stock decreases by one each time, success page works and the purchase button disables when stock is equal to 0.

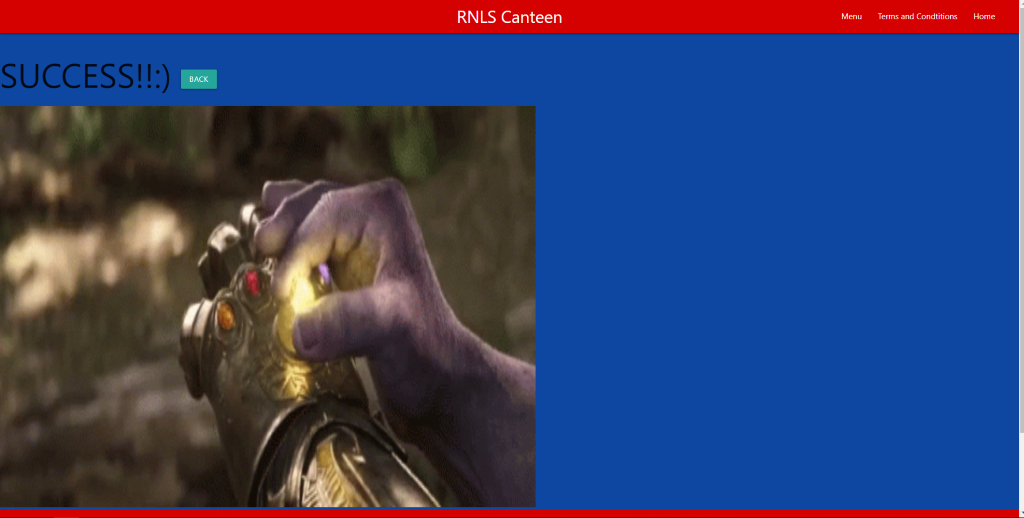
Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Had an issue where my titles for the food was being linked to the success page but was just a non-closed <a> tag.

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

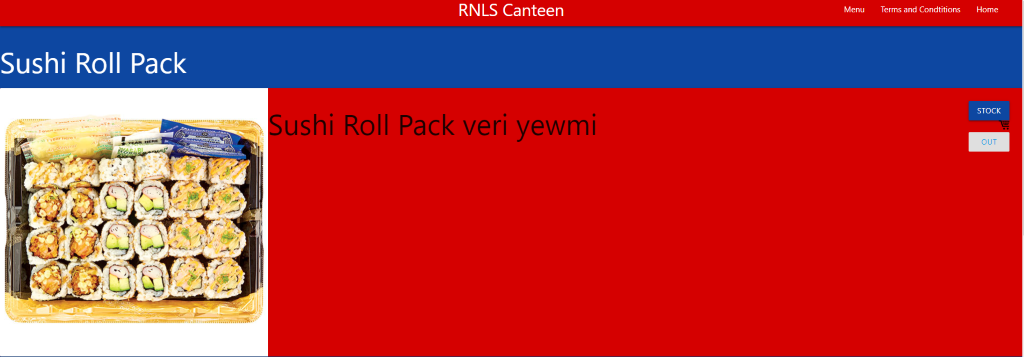


Purchase success page working

The button to purchase disables when the stock goes to 0.

Task 14 : Evaluation

*How did your version turn out*

Good, everything worked.

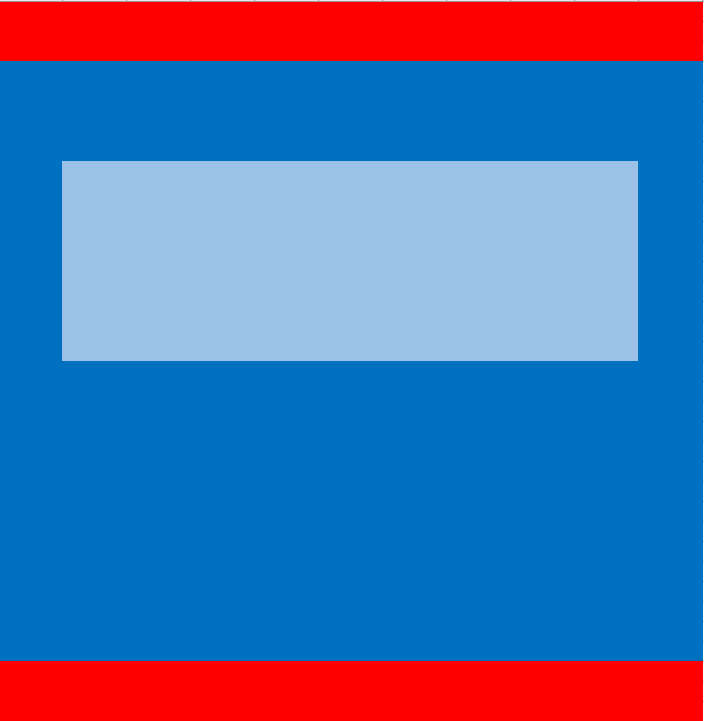
**Ver 5.0**

Task 0 : Explain what you are doing/ going to accomplish

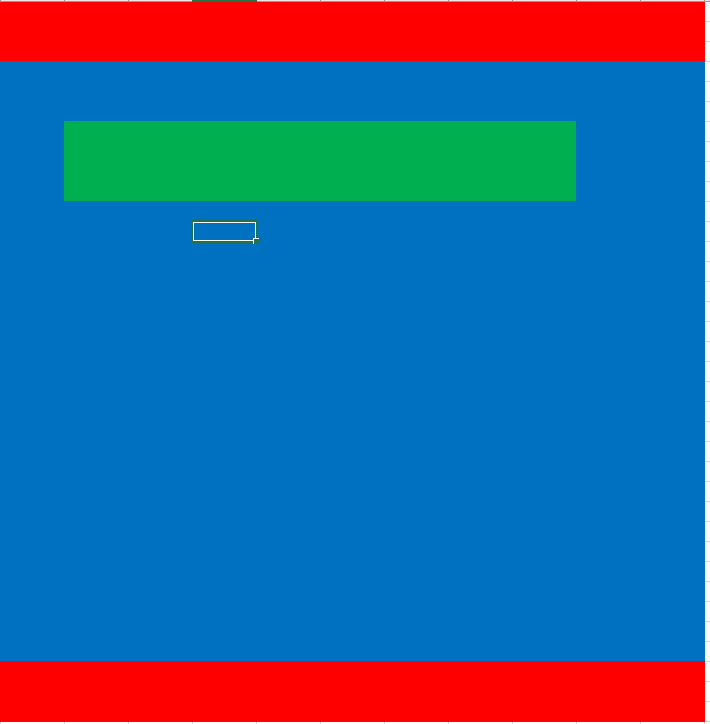
In this version I will add a restock function along with the table that displays all the prices and stock levels of the food at once.

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*



This is my table page, the light blue represents where the table will display the data



This is what my restock page will look like, the green represents the input field.

My restok success page will be the same as my purcase success page.

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Canteen\_food, contains the \_\_init\_\_ function which creates my objects

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

In this version I will need to display the stock, table and success page when you succesfully restock.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

They will be able to add stock.

Task 5: Identify any constants or existing data if required

Test data

* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12

Ham and Cheese Sandwiches - 4

As well as stock levels and price

Task 6: Identify indexed data structures

Canteen\_test

Contains my test data, as well as the stock levels, price, description and name.

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

Add stock when the user restocks a particular food item

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

Def restock\_page(): - this will match up the food id so the stock gets added to the right food item.

Def restock\_success(): - this will get the number inputted into the website and use it to add to the stock variable

Def table(): - will contain the dictionary will all the food data so I can display all stock levels at once.

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

At route (“/restock<food\_id>”)

at view(“restock”)

Define restock\_page(recieve food\_id):

Set food\_id to the integer of (food\_id)

Set found\_food to none

For food in canteen\_test: DO

If food.id is equal to food\_id: DO

Set found\_food to food

Set data to dictionary(set food\_list to found\_food)

Return data

At route(“/restock-success/<food\_id>”, set method to “POST”)

At view (“restock-success”)

Define restock\_success(recieve food\_id):

Set food\_id to the integer of (food\_id)

Set found\_food to none

For food in canteen\_test:

If food.id is equal to food\_id: DO

Set found\_food to food

Set data to dictionary(set food to found\_food)

Set restock to request.forms.get(“restock”)

Set restock to the integer of (restock)

Set found\_food.food\_stock to found\_food.food\_stock + restock

At route (“/table”)

At view(“table”)

Define table():

Set data to dictionary(set food\_list to canteen\_test)

Return data

In table, restock and restock success

%include('shared/header.html')

%include('shared/navbar.html')

%include('shared/footer.html')

In table page

%for food in food\_list

%end

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

Success page for restock – user needs to know what is happening

Have added some more comments in my python for readability.

Made sure user can only enter numbers in restock form, helps the user and code work better.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected outcome | Actual outcome | Fix |
| 1 | Add 1 to stock | Adds 1 to stock | N/A |
| 0 | Add 0 to stock | Adds 0 to stock | N/A |
| 100000000000 | Add 100000000000 to stock | Adds 100000000000 to stock | N/A |
| 99999999 | Add 99999999 to stock | Adds 99999999999 to stock | N/A |
| -50 | Takes 50 off stock | Takes 50 off stock | Added an min in HTML |
| Combination of letters | error | Error, there is a built in error that says input must be a number | N/A |
| symbols | error | Built in error, says input must be number | N/A |

Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

At first I was going to have a seperate page for restock instead of running it off the food page. This turned out to be too difficult as I could not get food\_id to pass from my webpage to my python server. To solve this I ran the restock off the existing card on the food page so that the webpage already has the food\_id as to create the card the ‘food’ page needs the food\_id. This turned out to work much better. I also had an issue with the method POST, my python server would not recieve what I was posting for some reason. But this was just because in this line

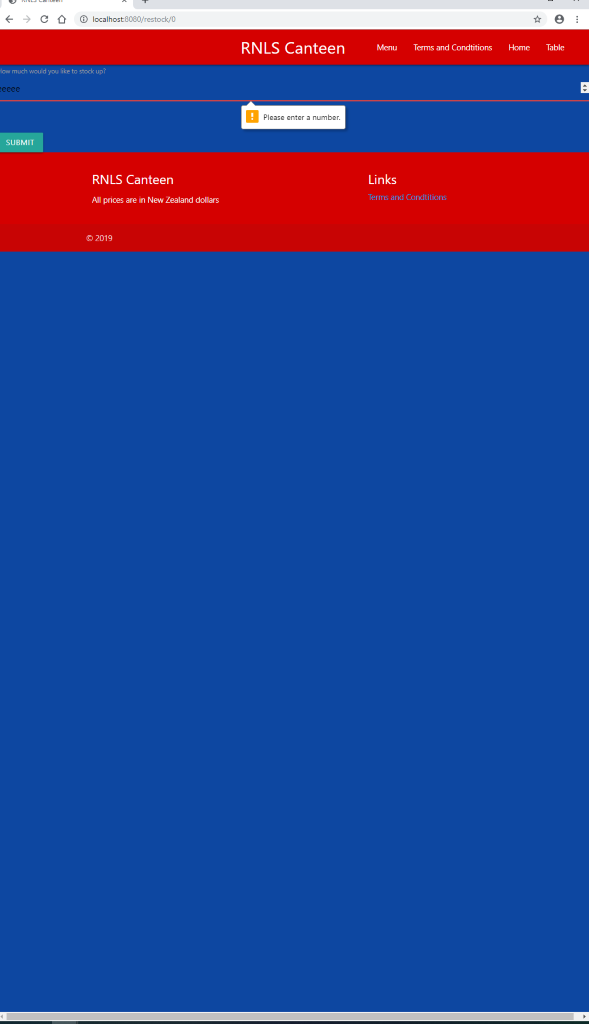
@route("/restock-success/<food\_id>", method="POST")

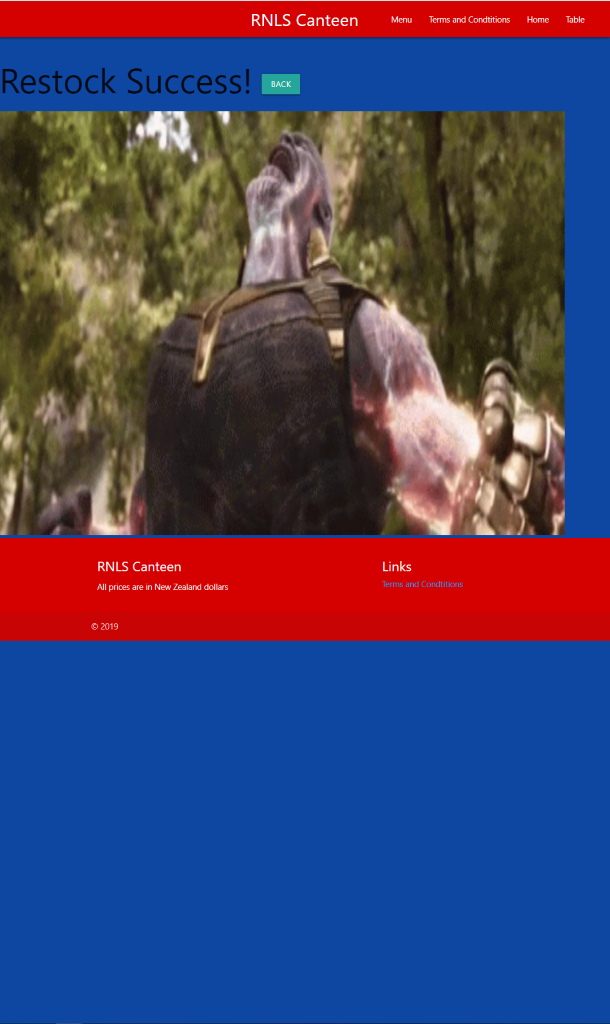
I had “/restock-success/<food\_id>” in “” marks and method = “POST” in ‘’ marks, so that was an easy fix.

Aslo added a min on my restock function so user cannot minus stock

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*





Task 14 : Evaluation

*How did your version turn out*

Very good, once I figured out how to change the restock so it works, was smooth running from there.

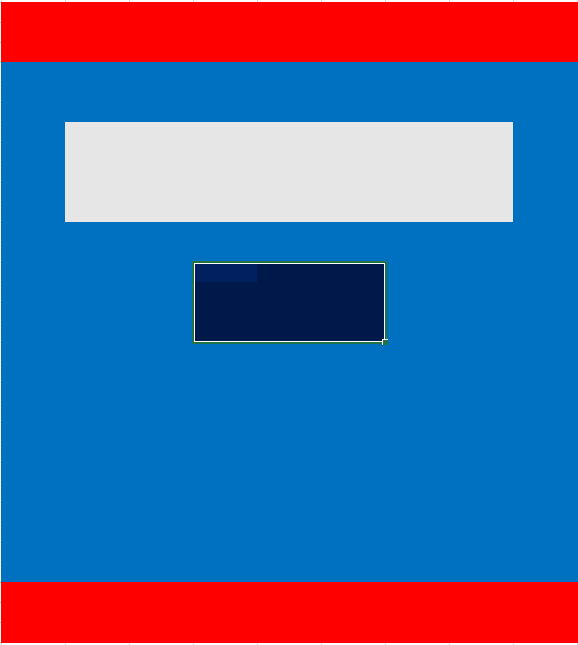
Ver 6.0

Task 0 : Explain what you are doing/ going to accomplish \

Add a variable which will count the total amount of food sold in a lunchtime, this will be attached to the bottom of my table page.

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*



Grey represents the existing table and dark blue represents the total food sold.

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

Canteen\_food, contains the \_\_init\_\_ function which creates my objects

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

Total food sold.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

N/A

Task 5: Identify any constants or existing data if required

Test data.

Task 6: Identify indexed data structures

Canteen\_test

Contains test data

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

Adding the individual food sold for each food item to get the total food sold.

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

In table page

% set total to 0

%set total to total + food.food\_sold

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

I have changed the maximum restock value from 500 to 50, 500 was too much for our small school of about 400, 50 is a much better number. More sensible.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

If the table displays the right amount of total food sold it works.

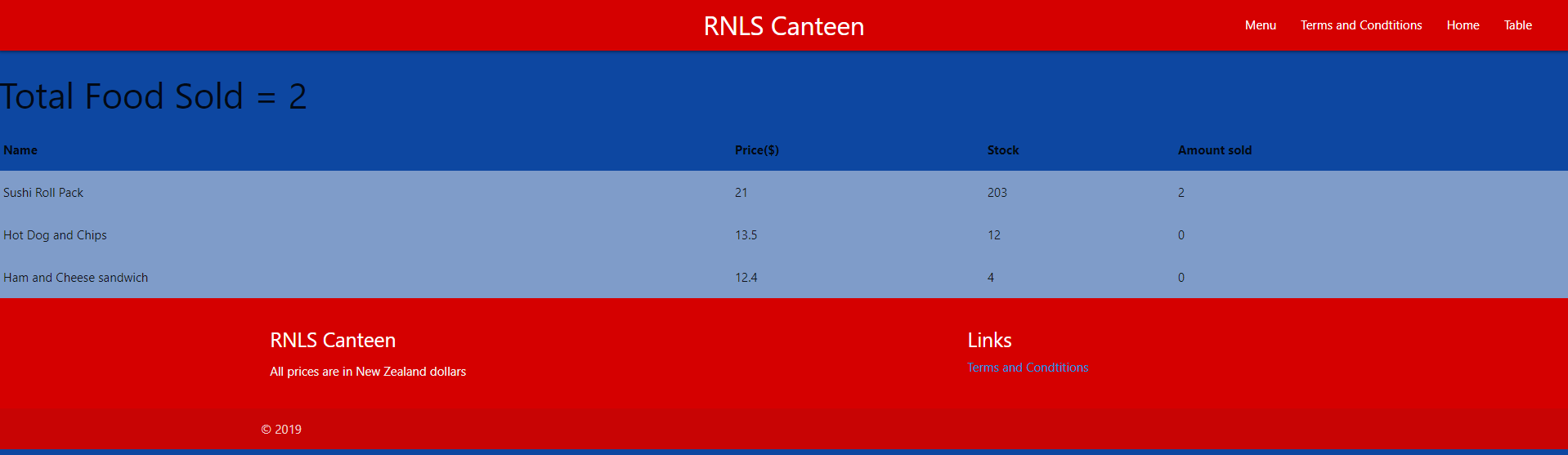
Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Had a small error where I was making a new variable to have the total food sold but I needed to add all 3 individual food sold together.

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*



Task 14 : Evaluation

*How did your version turn out*

Good, didn’t have any huge errors except when adding the individual food\_sold together to get total food sold. I was trying to get my function to add 1 to total\_foodsold each time the user purchased a food item. But this didn’t work so I added each individual food\_sold together.

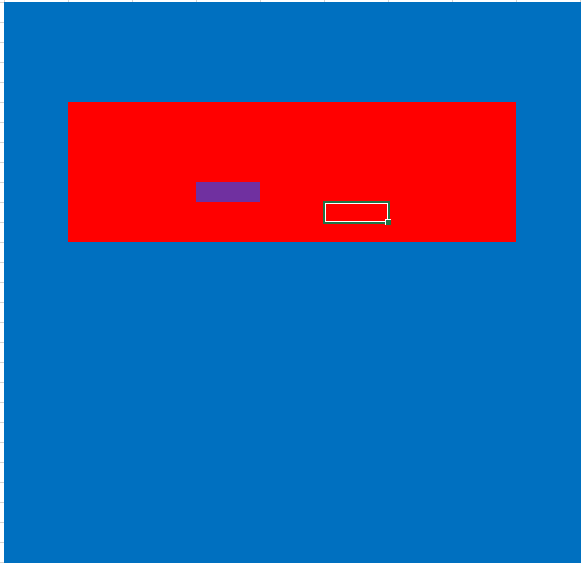
Ver 7.0

Task 0 : Explain what you are doing/ going to accomplish

All relevant fixes to get a better mark. Specifically testing and any relevant implications. Also redesigning the restock function.

Task 1: Sketch interface design

*Draft a rough design for the interface that allows the user to trigger functionality in task 1, while also annotating where the information in task 2 will be displayed. Create another sketch listing the interface widgets used to create the interface.*



Red is food item and purple is food restock button. There will be a field for each food item.

Task 2: Identify any classes required

*Explain what the class will represent, plus listing what information will be stored in the class and any functions the class will have.*

N/A

Task 3: Identify information to be displayed

*What information will the interface need to display to the user?*

The restock page, with food items and button for restock.

Task 4: Identify user inputs

*What program functions can the user trigger through the interface?*

Restock value, cannot be more than 50.

Task 5: Identify any constants or existing data if required

Test data

* Sushi Roll pack - Starting with 5
* Hot dog and Chips - starting with 12
* Ham and Cheese Sandwiches - 4 in stock

Task 6: Identify indexed data structures

N/A

Task 7: Determine what calculations are necessary

*Write out the calculations the program will have to compute.*

Adding stock when the restock function is used.

Task 8: Develop a modular structure for your program

*Describe any functions that the computer program will have, identifying any sub-functions where required.*

N/A

Task 9: Define the functions identified

*Describe the functions for both the main program and any classes in terms of input and/or output where required. You may choose to do this with flow charts or pseudo-code (not Python code!). Add in additional steps or explanations using sequential, conditional, iterative statements where required. Identify global and/or local variables.*

N/A

Task 10: Address any relevant implications such as usability, functionality, legal/ethical requirements.

Added comments, variable names, success page. Peer feedback

Have added more comments in my python code.

My variable names are all named accordingly so any one who looks at my code can see which variable has what.

My success page works for the restock and purchase page so the user knows what is happening.

Peer feedback – got a class mate to navigate around my website and give feedback, he said very good, very understandable and easy to navigate around.

Task 11: Document test cases for testing the program

*Document any testing that can be used to test your program. If any input is inputted using the keyboard, describe the expected input, plus any exceptional, boundary or invalid cases.*

Task 12: Refine the plan

*Note any modifications here when iterating through the development cycles.*

Got some peer feedback, they said I should move the restock function to a different page. Also to remove the home button from the navbar. So I will create a new page, similar to the menu page for the restock function. Colour matching

Task 13: Document testing

*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

Task 14 : Evaluation

*How did your version turn out*

Good, all fixes worked.