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

+stock page, set up the contents with their stock and add an add stock insert tab that allows the user to stock up as much as they want, as well as a simple stock page that shows the amount of item there are, how many have been sold, and the cost of the items.

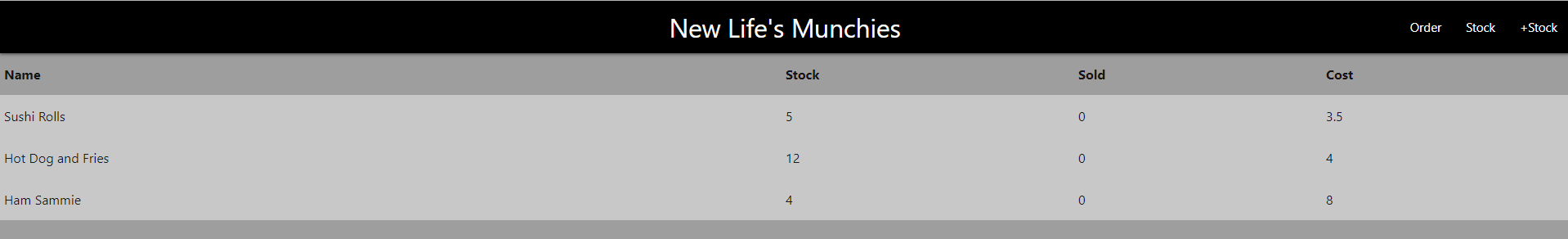
Task 1: Sketch interface design

*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.*

Plus stock page:



Stock 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.*

Amount

Task 3: Identify information to be displayed

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

The +stock tab on the nav\_bar, a list of the food items with the cost, the stock and how many have been sold. It will also have a add button where the user can add to the stock of the food item, when clicked it will take them to another success page that will display a success message and a back button, that when clicked will take them back to the +stock page.

Task 4: Identify user inputs

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

The +stock tab on the nav\_bar

The add button

The user input number value

The submit button

The back button on the success page

Task 5: Identify any constants or existing data if required

Food.id

Task 6: Identify indexed data structures

Contents / contents\_list

Task 7: Determine what calculations are necessary

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

Stock = stock + 1

Task 8: Develop a modular structure for your program

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

Plus\_stock – this is the page where you can add stock to the food items

User\_input – This will take a value given from the user and store it in a variable

Add\_success – This will add the value from the user\_input to the stock of a food item

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.*

**PYTHON - Pseudocode**

AT ROUTE (‘/plus\_stock’)

AT VIEW (‘plus\_stock’)

PROGRAM plus\_stock

SET data TO dict (contents\_list EQUALS contents)

RETURN data

AT ROUTE ‘/add\_success/ <food.id>

AT VIEW ‘add\_success’

PROGRAM add\_success (food.id), METHOD EQUALS ‘POST’

SET amount TO request.forms.get(‘amount’)

SET amount TO int(amount)

SET Food.id TO int(food.id)

SET found\_food TO NONE

FOR food IN contents

IF food.id EQUALS food.id

SET found\_food TO food

SET data TO dict(food EQUALS found\_food)

Found\_food.food\_stock PLUS EQUALS amount

RETURN data

AT ROUTE ‘/user\_input/ <food.id>

AT VIEW ‘user\_unput’

PROGRAM user\_input (food.id)

SET Food.id TO int(food.id)

SET data TO dict(contents\_list EQUALS contents)

SET found\_food TO NONE

FOR food IN contents

IF food.id EQUALS food.id

SET found\_food TO food

SET data TO dict(food EQUALS found\_food)

RETURN data

RETURN data

AT ROUTE ‘/stock’

AT VIEW ‘stock’

PROGRAM stock

SET data TO dict (contents\_list EQUALS contents)

RETURN data

**CSS (python):**

%include('shared/header.html')

%include('shared/nav\_bar.html')

% FOR canteen\_content IN contents\_list:

%END

% for canteen\_content in contents\_list:

%END

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

The user will need to click the +stock tab, then proceed to click the add button at the end of every item in the foods list, for whatever they want to add stock to. They will then need to input a number for how much they want to stock up at once (i.e 26). There will also be a success page and the user will have to click the back button to get back to the plus stock page.

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 if I can add stock if the food item is out if stock. Test if adding will work normally i.e 5 stock to 6. Test if will go above 100 (it should be able to), test if I can add negatives, test if I can add 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Test case | Expected outcome | Actual outcome | FIX |
| 0 | Shouldn’t work | Didn’t work | NONE |
| -50 | Shouldn’t work | Didn’t work | NONE |
| Add if there is 0 stock | Should work | Did work | NONE |
| +99999999999999 | +99999999999999 | +99999999999999 | NONE |
| +1 | +1 | +1 | NONE |

Task 12: Refine the plan

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

I have changed the line **SET found\_food.food\_stock TO found\_food.food\_stock + 1**

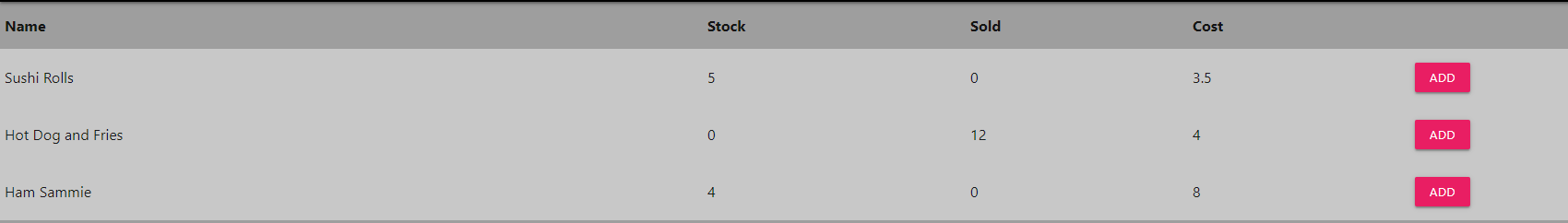
To “**found\_food.food\_stock += 1**” (This is a more robust way of code)

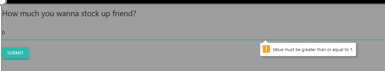
I have now added a user input page, where the user inputs a value and the values name is amount. With all that I have changed it to **found\_food.food\_stock += amount**

I will probably change up how the success page looks later on.

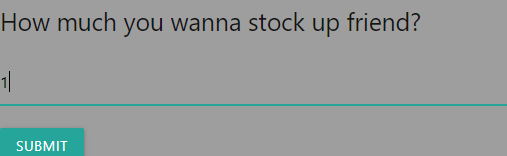
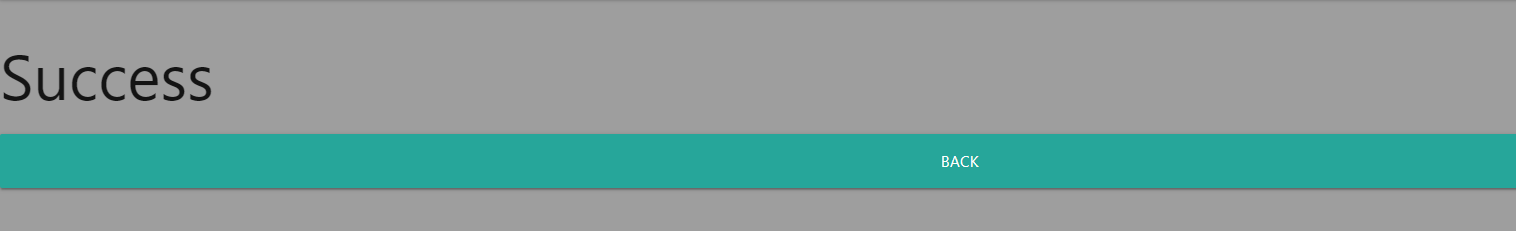
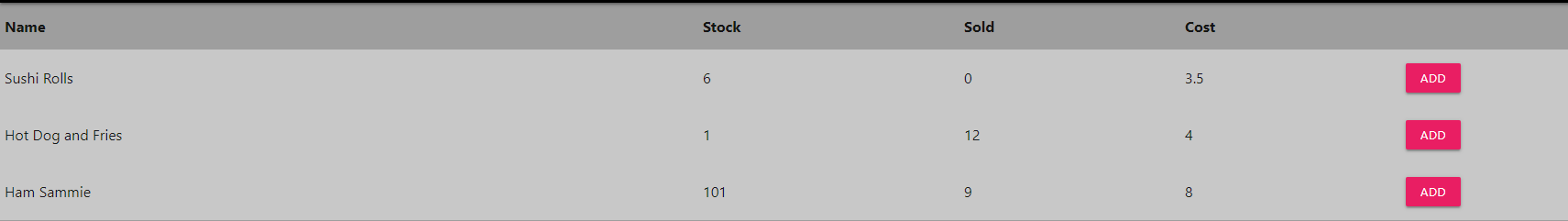
Task 13: Document testing

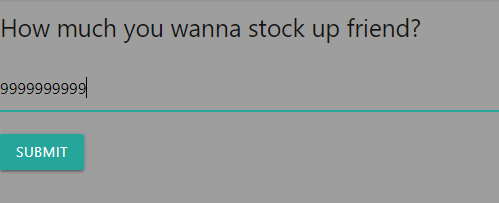
*Show screenshots of your program working with descriptions of each image. These images should test the tests cases listed above.*

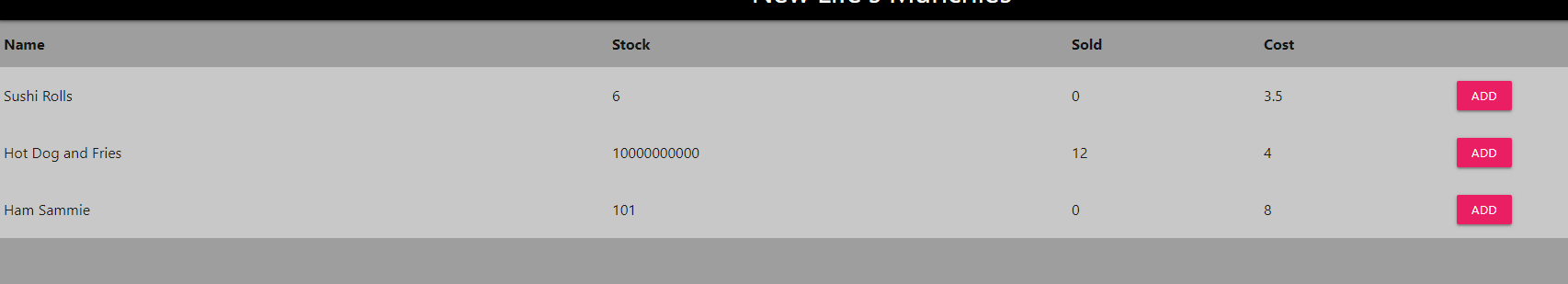


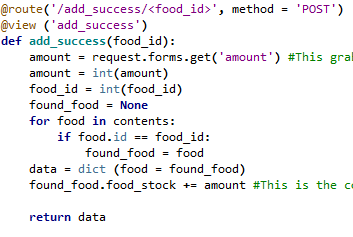




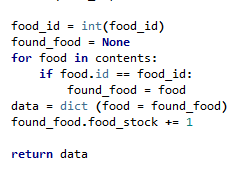
  
 







-------------------------------------------------------------------------------------------



Task 14 : Evaluation

*How did your version turn out*

This version turned out ok but was a process. At first I had made it so my plus stock page could only stock up one at a time, after I had done this, I realised the criteria states it needs to be restocked up as much as the user wants. So I decided to stick to the same version (I.e ver 4.0. ver 4.1, ver 4.2 ect..)

To get it so the new stock value was as much as the user wanted, took me a few periods, there were many times I had gotten stuck by the simplest thing, for example a full stop instead of a underline. I did need help from the teacher once but afterwards I understood where I went wrong and was able to finish what I needed.