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

In this version I will be able to increase the amount of stock of a specified item by a given amount.

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.*

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.*

Task 3: Identify information to be displayed

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

I will need a button to click if you want to input stock and a form to ask how much stock you want to add.

Task 4: Identify user inputs

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

Task 5: Identify any constants or existing data if required

Task 6: Identify indexed data structures

Task 7: Determine what calculations are necessary

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

Task 8: Develop a modular structure for your program

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

Restock, it will be my function for my restock page.

Restock Success, it will be my restock success page

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.*

ROUTE restock METHOD post

VIEW restock

PROGRAM restock

Make data a dictionary of my items

RETURN data

ROUTE restock\_success PASS item id

View restock\_success

PROGRAM restock\_success

item\_id = int(item\_id)

found\_item = None

FOR item IN items

IF item.id IS item\_id

SET item TO found\_item

SET data TO Dictionary (SET item TO found\_item)

Not sure what to do for rest of code. Maybe make stock, stock + stock\_add

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

I made my page the school colours to match the theme of other school websites. I made my buttons stand out so it is easier to see where they are. My restock and restock success pages are visually appealing and there is a clear way to return to my restock page from my restock success page in the form of a button. I also made sure my wording makes sense so that people visiting my website can understand how to navigate easily and what each button does.

You can clearly see where the information for each of my items stops and another begins. This is to improve the visual clarity and make my page easier to understand. It also is important for people to know what information relates to each item so that they restock the intended item.

I have included comments in my code so that when someone looks at it, they will understand what each section of code does.

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.*

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Expected Result | Actual Result | Fix |
| 10 | +10 to stock | +10 to stock | N/A |
| -10 | Unable | -10 to stock | Set min to 0 |
| 1000000 | Adds 1000000 | Adds 1000000 to stock | I am setting a max of 10000 |
| Boy | Doesn’t input anything | Doesn’t input anything | No fix needed. |

Task 12: Refine the plan

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

Added:

restock\_add = request.forms.get('restock\_add')

restock\_add =int(restock\_add)

found\_item.stock = found\_item.stock + restock\_add

Task 13: Document testing

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

Could not add int and NoneType.

<input id="restock\_add" type="number" class="validate">

As you can see I have type specified as number. However I do not have a name for this input so when attempting to pass to my python page it cannot attach the decorators to a variable name.

Fix: <input id="restock\_add" type="number" name="restock\_add" class="validate">

Added a min of 0 to restock. This prevents people from adding a negative number of stock. Added a max of 10000 to prevent crazy extremities.

Task 14 : Evaluation

*How did your version turn out*

This iteration of my code worked out as intended. I created a new page called restock which contains information relating to stock amounts of each of my food items. You can increase the stock of each item by a specified amount. I found this iteration of my code to be the hardest as I used techniques I had never used before. In testing my code I made some changes to my program to fix the issues.