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

In this version I will write code which allows my user to purchase more than one of each item at a time.

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?*

An input for how much to increase stock by and how much stock there is currently

Task 4: Identify user inputs

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

How much to increase stock by.

Task 5: Identify any constants or existing data if required

I will be writing this code in my existing function Purchase Page

Task 6: Identify indexed data structures

Task 7: Determine what calculations are necessary

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

Stock – how much has been purchased

Task 8: Develop a modular structure for your program

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

I am going to be using my purchase page function

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.*

SET Purchase\_amount TO the input of my form on my webpage (purchase\_amount)

Make sure it is an integer

Found item stock – purchase amount

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

I need to make sure that the input box for how many items you want to buy is in an obvious place and that it is not in an unexpected colour or wording. My page is school colours to match with other school websites.

A criticism I had was that it was difficult to get back from my purchase success page. I fixed this by adding a button in the middle which stands out in a bright red. This links back to my purchase page so the user can easily get back.

I have put comments in my code so that when I or another looks back on my code, they will be able to understand it.

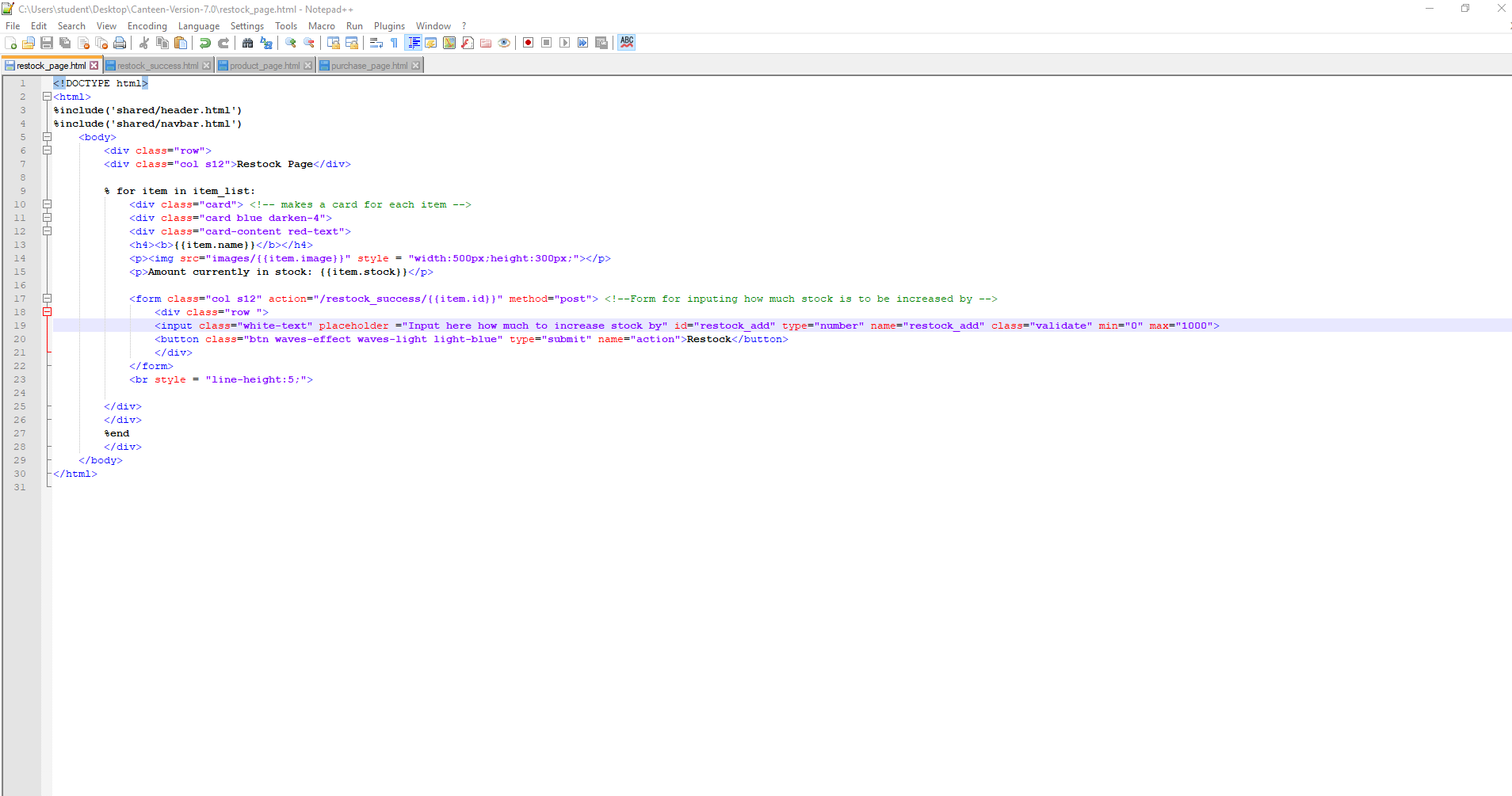
Task 11: Document test cases for testing the program

*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 |
| 1 | Takes away 1 | Takes away 1 |  |
| -2 | Doesn’t let you | It added 2 stock | Set min to 1 |
| 15(more than current stock) | does not work as is more than current stock | Negative stock | Set max to {{item.stock}} |
| hello | nothing | nothing |  |
| Nothing. | Nothing happens | Throws an error | Added attribute “required” |

Task 12: Refine the plan

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



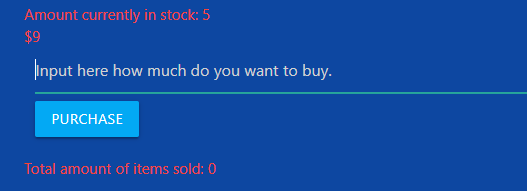
input class="white-text" placeholder ="Input here how much to increase stock by" id="restock\_add" type="number" name="restock\_add" class="validate" min="0" max="1000"

This used to be max 10000 but since this is a school canteen I thought it would be too much.

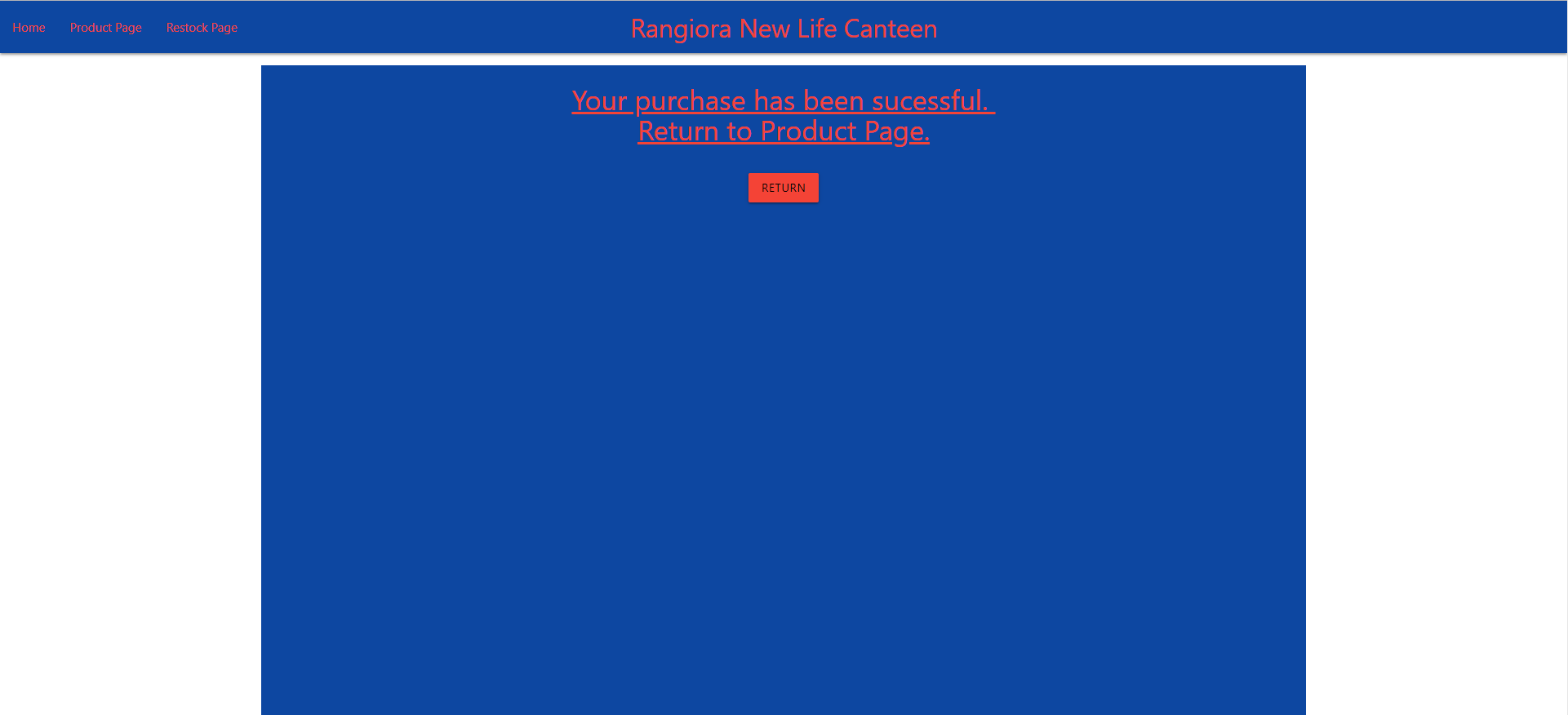
Hector wondered why my minimum was 0. This was a good point as there is no point in purchasing 0 stock. To resolve this I changed min to 1.

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 box to input the amount of an item you want to buy is obvious as is the button.



This shows the purchase went through successfully and has an obvious way to return to the restock page.

Task 14 : Evaluation

*How did your version turn out*

In this version of my code I added the ability to purchase multiple items at one time. I had some issues with my code from previous versions so I went and fixed those. One of these is when I inputted nothing into my form, it would throw an error. I fixed this by adding an attribute called required. This means that the field must have an input otherwise it will not allow it.

I also moved restock\_add = request.forms.get('restock\_add') to the top of my functions. I did this to make my code more clear. I also added blank lines to parts of my code to separate parts of my code.

This version worked as intended and this is the last version of my code.