**Appendix 1: Planning Guide**

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

In this version I am going to change my restock function so that it is able to add more than 1 stock 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.*

N/A

Task 3: Identify information to be displayed

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

A box asking how much a user wants to add will need to be displayed

Task 4: Identify user inputs

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

User will input a number for the amount of stock they need to add

Task 5: Identify any constants or existing data if required

N/A

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 the desired restock amount to the current stock

Task 8: Develop a modular structure for your program

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

No new functions needed

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.*

quantity BECOMES request.forms.get 'quantity'

quantity BECOMES integer ‘quantity’

found\_item.stock BECOMES found\_item.stock PLUS quantity

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

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.*

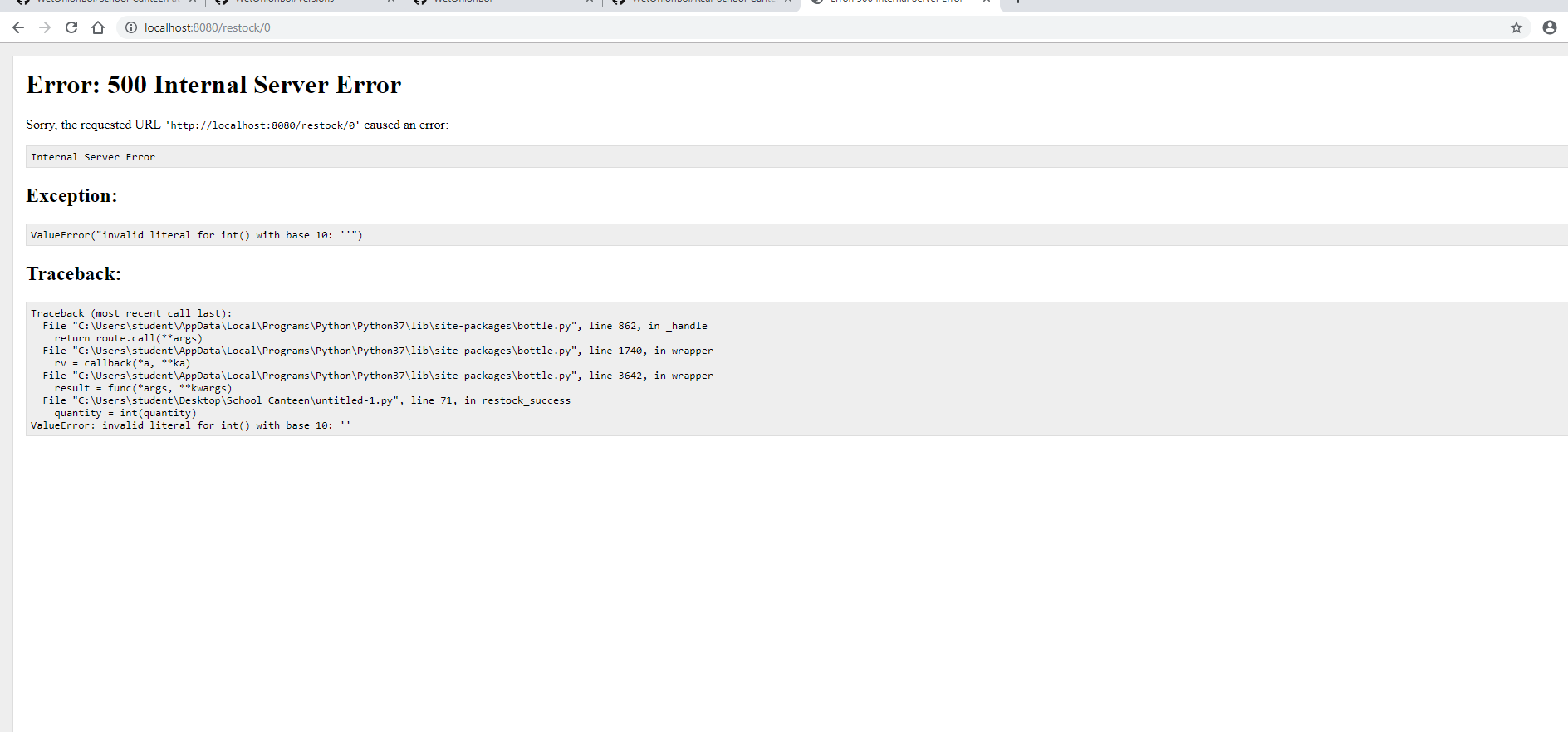
*I am going to test the function by loading the page and inputting a large number to see if it adds it to the stock. I will also input a letter and a negative to see if it will minus from stock.*

Task 12: Refine the plan

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

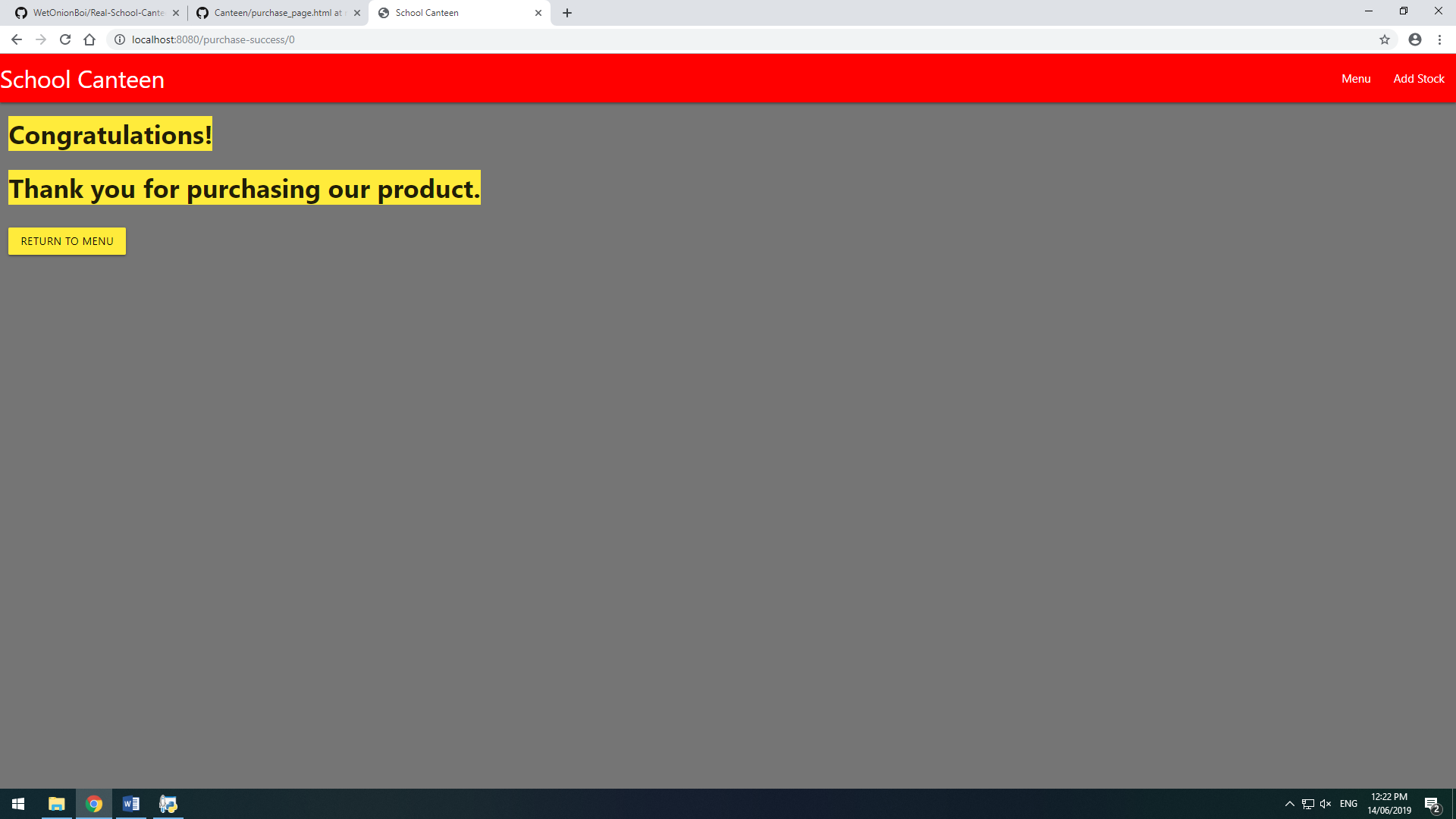
None were necessary

After revisiting this version I realised that if the user inputs nothing it throws an error



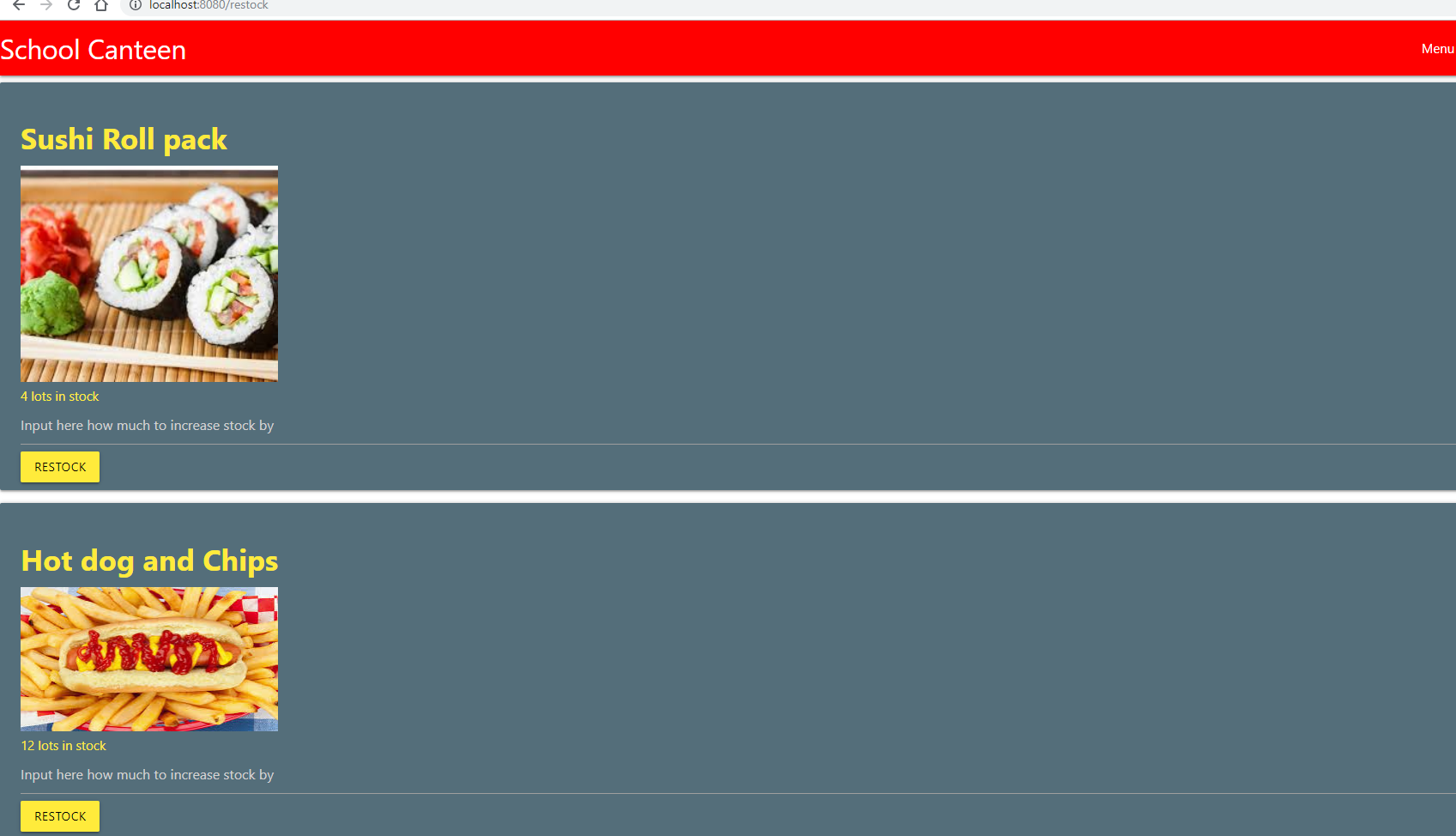
I am going to type required into my html input field to make sure that the user has to input something

Second time revisiting I changed the html too look cleaner and cleaned up my site.

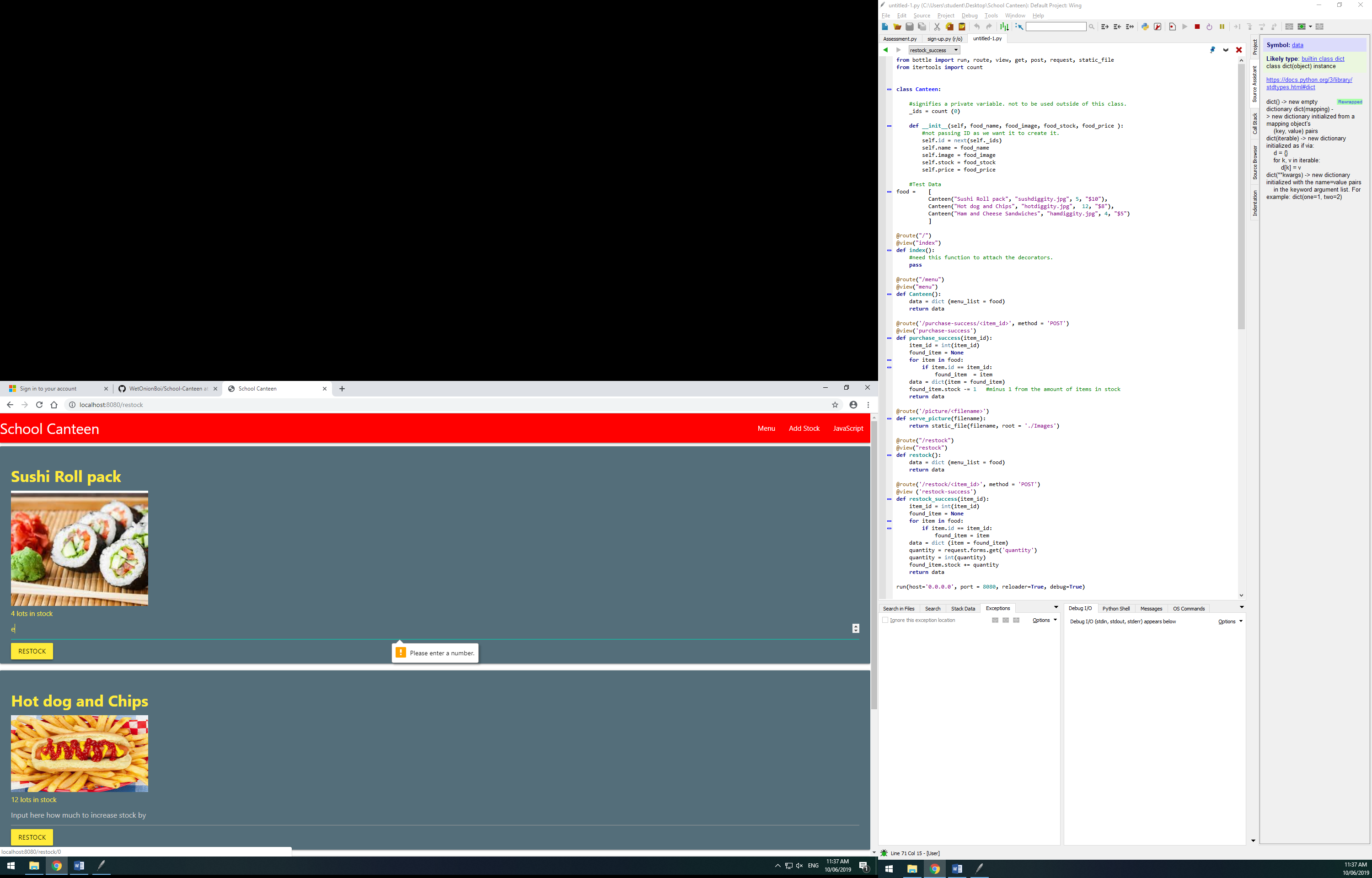


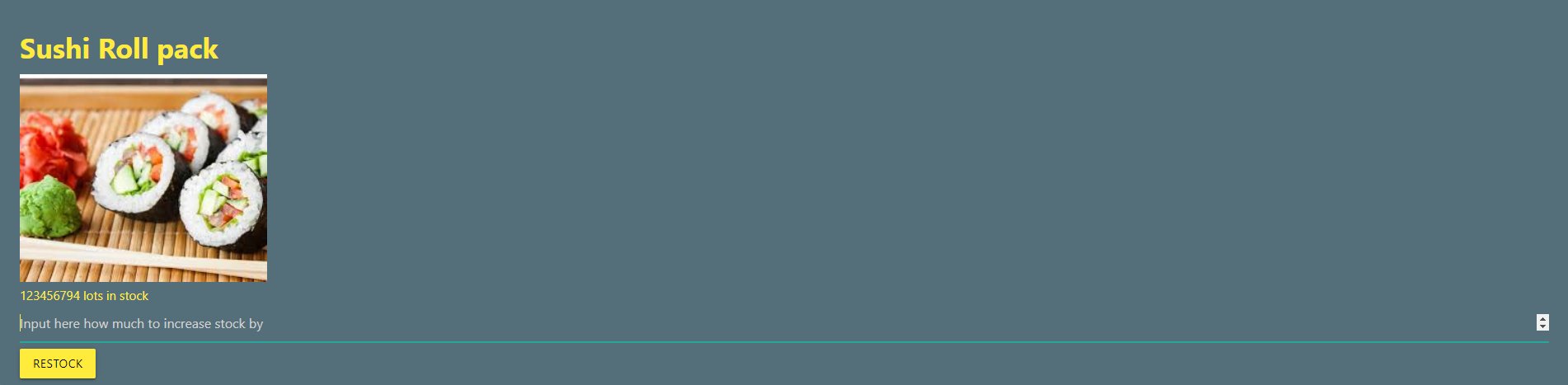
Task 13: Document testing

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



Wont allow any letters to be inputted apart from e although I think that this is due to the fact that it is a mathematical term.





Task 14 : Evaluation

*How did your version turn out*

This version works how I wanted it to in task 0 and I didn’t need to make any modifications to my plan. In the next version I will add an admin login so that admins only will be able to restock. I will also add a function that allows users to see how many of each item was sold.

Reevaluation – After fixing the bug that was coming up where users can click restock straight away without inputting anything the site worked well.