**Implications**

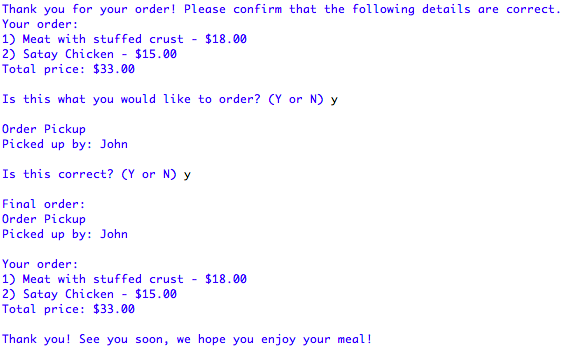
An implication that I considered while making my program was usability. The user should be able to navigate the program freely and feel in control of what is happening. If they make a mistake while entering information, it should be easy for them to recover and carry on. The program should be consistent throughout, and let the user know what is happening at all times.

There are multiple ways that I addressed this implication. For example, for user freedom I implemented a menu-based system for my program so that the user can choose what is happening and when based on what suits them. If they only need to see the menu once, then they only have to show it once. If they want to order one pizza and take it straight to check out, they can. If the user is indecisive and likes to look at their order a lot and make a lot of changes, and possibly completely start over at some point, they can.



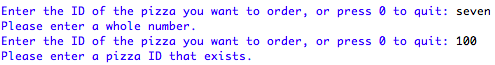
*Screenshot showing the menu prompt with all the options for the user.*

I also took measures to recover from any errors if they happen. If the user enters something that they do not want to order, they have the option to remove it right away by editing their order and removing the unwanted pizza. Also, at the end of the program, there is a confirmation process for both the pizza order and the delivery/pickup information to make sure that the user is not placing their order unless they are 100% happy with it.



*Screenshot showing the confirmation process at the end of the program.*

If the user entered something that the program could not work with – for example, a string when an integer is required, the program can cope with this and ask them to enter again rather than crashing.



*Screenshot showing the error control messages for the pizza ordering process.*

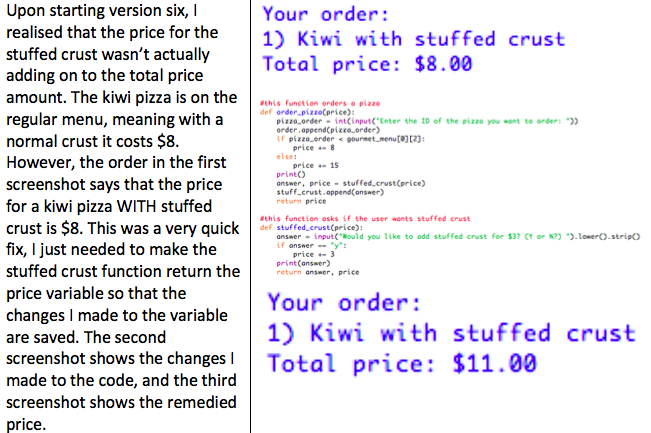
One way that I ensured consistency throughout my program was through the use of functions. There is a lot of code that gets repeated, for example the output of the user’s order. Not only would it be a waste of time to write all of that code out multiple times, but there is the risk that I write out something wrong and cause an inconsistency throughout the program. Through the use of functions, I can be sure that the code that I am calling is the same every single time and the user is not taken by surprise.



*Screenshot showing a function that gets called many times.*

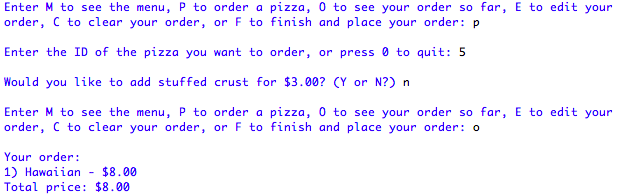
Another implication that is important to consider is functionality. While it’s important for the program to be easy to use and look good, it’s even more important that it works as intended and fulfils its purpose. Otherwise, there’s no point having a program at all.

The purpose for this program was to enable the “Heavenly Pizza” pizzeria to operate contactless during the Covid-19 pandemic so that they could still make money and not go out of business. So, the main function of the program is to allow the user to buy pizzas. While I can’t actually put through the order and deliver a pizza to the user because the pizzeria doesn’t exist, I created a program that addressed the purpose and works as I intended it to. At every point in development, as I coded different components of the final product, I tested to make sure that it was working as expected before moving on. This helped me to keep track of what I was doing and ensure that, above all else, my program is functional.



*Screenshot showing an entry into my testing and evidence log that shows how I noticed a problem in the functionality of my program and fixed it before doing anything else.*

Because the program is entirely text-based, there is not a lot that I could do to address aesthetics or accessibility. However, one thing I did that I think addresses both of these implications is using blank print statements to space out the output in my program. For aesthetics, this utilised negative space and meant that not everything was all clumped together. For accessibility, this made the text easier to read and meant that the user didn’t get confused by being overloaded with a large block of text.

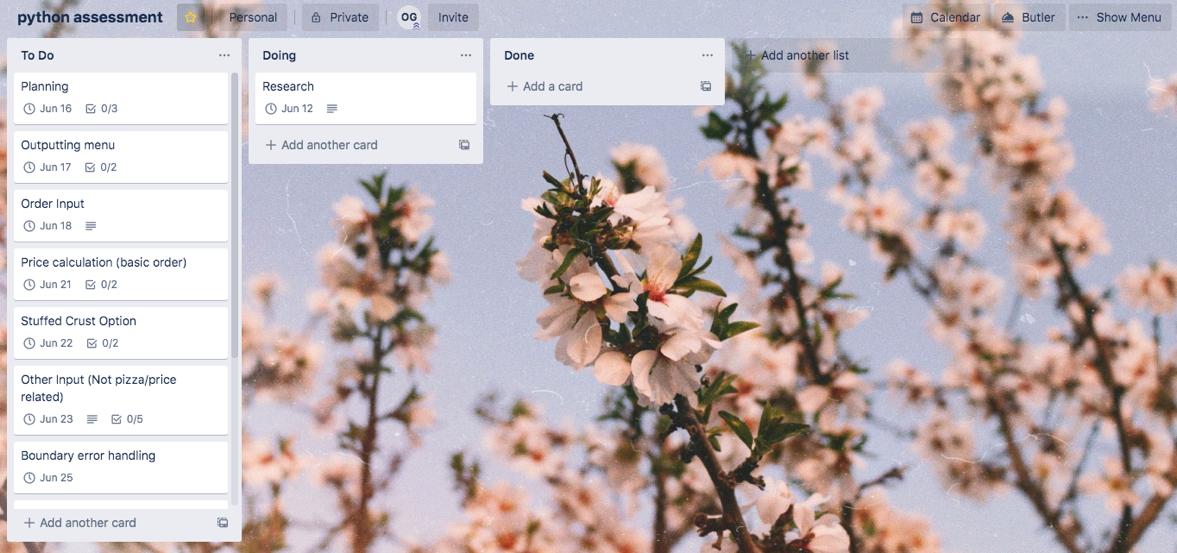


*Screenshot showing the spaced out text in the program.*

**Discussion**

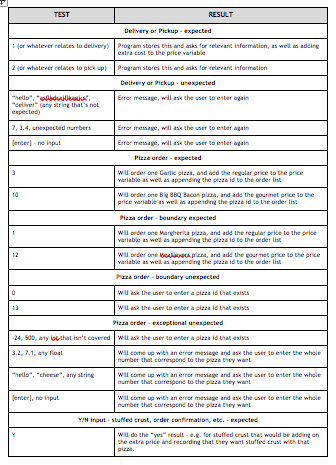
The information from planning, testing, and trialling components of my program helped me to develop a high-quality outcome.

During my planning process, I created a Trello board and decomposed the different aspects of the program into “cards”, with separate descriptions and/or checklists. For each card, I set a deadline to help me keep track of my time management for this large project. By decomposing the final outcome into manageable chunks, I was able to focus on getting one aspect of the program at a time to work. For example, in version one (second entry in the testing log) I just set up the menu and made sure that it was outputting correctly because this is essentially the backbone of the program. The user can’t order pizzas if they don’t have any pizzas to order. Ensuring that each portion of the program is developed to a high quality increases the overall quality of the final outcome.

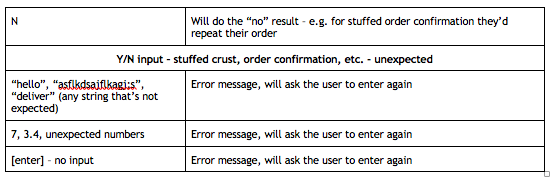


*Screenshot showing my trello board at the very beginning of the project. On the left, in the “To Do” column, are all the different components of the program with individual due dates and tasks within them.*

After completing each of these versions, I would test to make sure that they were working as expected. With each version entry in my testing log, I would show not only how I coded that component, but also how I tested it. In addition to this, after adding error handling measures I followed my testing plan to test every single possible way to try and break the program so that I was sure it was of sufficient quality.



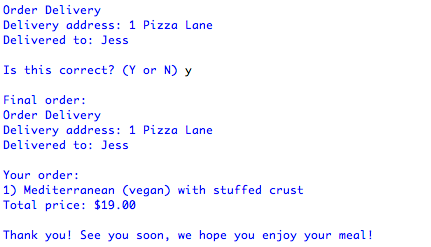
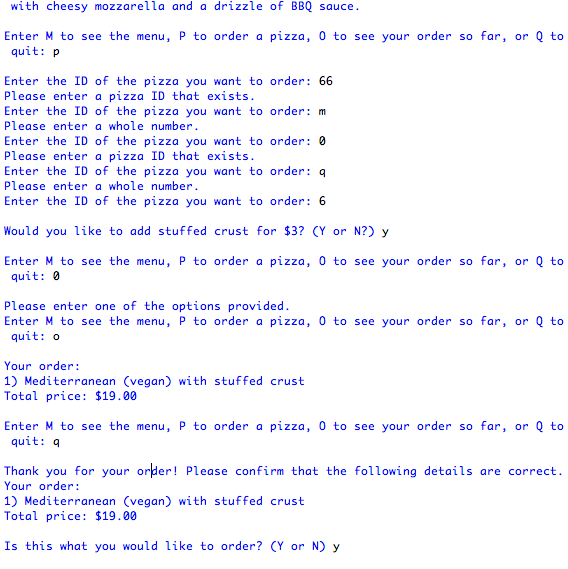
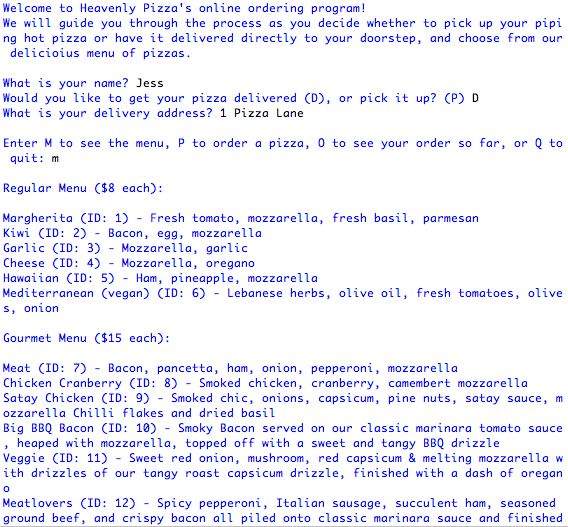
*Screenshots left showing the testing plan, it can also be found at the bottom of the planning document.*

This was all testing that I performed on my program during the development process that helped me to create a high quality final product, because I could identify any problems with the way that my program was functioning and fix them.

However, it is not only the testing that I performed that was useful in the development process. I’m used to my program and the way that it works, and I know how it is meant to be used. This means that I cannot see some of the problems that may annoy or seriously hinder another user.

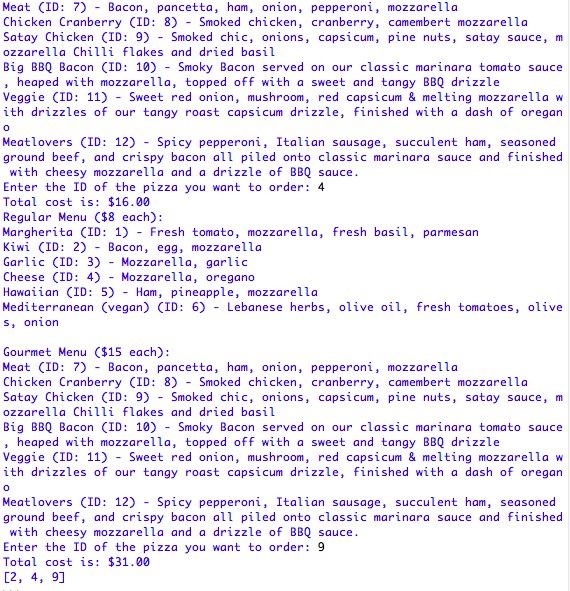
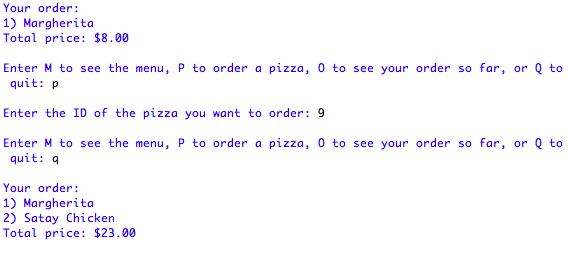
I got other people to perform some end-user testing in the middle and at the end of the development process. First, I will talk about the testing at the end. I gave my program to some different testers and asked them to try it out, not telling them anything other than the fact that it’s a “pizza ordering program” so that they can give me their completely honest, blind opinion.

From this testing, I received some very good feedback and from this I made some minor and some major changes to my program. I have gone into more detail about these changes in my Testing and Evidence Log. This user testing helped me to develop a high-quality outcome, because without it I would have never made these changes that were positive and increased the usability of my program.



*Screenshots showing one of the tester’s run-throughs.*

Finally, trialling was another aspect of the development process that helped me to produce a high-quality outcome. Upon finishing version three of my program, in which I added the capability for the user to order more than one pizza, I noticed various problems that I would find very annoying as a user. In the version three entry in my testing log, I go into much more detail on the specifics of these problems. So, I created a version four which used a menu-based system which addressed these problems.

*Screenshot left shows the third version of the code, screenshot below shows the improved fourth version.*

I also got an end-user tester to trial these different versions and tell me which one they preferred. They said that they preferred the new version (version 4), which was the one that I thought was better too, so that is what I went with. It has changed a bit since then, but in the final version the basis of the system is still the same. This helped me to create a high-quality outcome because this was an important decision that shaped the flow and function of the program, and it would be unwise to make it all on my own.

Specific advice that I got from the tester is in the Testing and Evidence Log in the Version Four entry.