|  |  |
| --- | --- |
| **Testing/Evidence Log** | |
| Pre-coding development:  Before anything else, I set up a Trello board as a way to keep track of my project management. This was what it looked like to start with. Then I did some research on similar programs, which can be found in the Research folder. While I was researching aspects of these programs, I noticed that they ask the user to enter the time they want to pick up their pizza or get it delivered. I talked to the stakeholder about this, and it was decided that this feature would not be necessary in the program that I am making. The second screenshot shows my board once I had finished my research and planning documents. The final screenshot shows the calendar of due dates for certain aspects of this project. As you can see, I was now running ahead of schedule. |  |
| In version one, I focused on getting the menu of pizzas set up and being able to loop through and print them all out. I felt that this was a good place to start, because everything centres around the pizzas in the program and I needed to make sure that it worked before I started on anything else. |  |
| In version two I focused on getting the input for the user to order a pizza. The user will enter the number that corresponds to the pizza that they want to order, and the program will store this and output the price of the pizza and the id that they inputted. At the moment, it only takes in the order for one pizza, but that will change later.  I tested this with a few different pizzas to make sure it was outputting the right price based on the menu the pizza was from. |  |
| In version three I added the capability for multiple pizzas to be ordered at once, and then calculated the total price of the order. The user enters the number of pizzas that they want to order, and each time the menus print out and the user enters what pizza they want. However, upon testing this, I realised a few problems. One was that the user had to decide how many pizzas they wanted to order from the beginning, before even being able to see the menu. Also, the large output of the menu happens every single time, which could be annoying for the user if they don’t want to see the menu again. So, I decided to try another way of presenting the output and get the opinion of a tester. |  |
| Here are screenshots of my trello board and trello calendar at this point in the development. Time-wise, I was a little under half way, and this time is represented pretty well by the amount of tasks in the done column. Also, as seen on the calendar, I was running a few days ahead of schedule. |  |
| In version four, I changed the input method to solve the problems I identified in version three. I used a menu based input method, so that the user was in control of when they wanted to see the menu, whether they wanted to see their order so far, etc. This way they don’t have to decide how many pizzas they want to order at the start – if they see the menu and decide they want to order an extra pizza, that’s fine. I got a tester to try both versions of the program, and got their opinion on which one was better, and what needed to change in the future. The feedback that I got was that version three (where they start with entering how many pizzas they want) was overwhelming because the long menu printed out every time. However, they liked the choice that the menu-based program provided, and how the menu did not print out unless they wanted it to. They gave the advice that I should change the output at the end after they quit, as it looked exactly like when they output during the order process. I have noted this, and I will change it when I get to changing the output at the end of the program.  One other thing that I changed is that, whenever the program is deciding whether a pizza is from the regular or gourmet menu, the comparison it performs is pizza id < 7. However, if another pizza was added to the regular menu, this would no longer work. It needed to be smaller than the first id in the gourmet menu, whatever that may be. So I changed this to pizza id < gourmet\_menu[0][2] (the third item in the first list). |  |
| As I was adding the stuffed crust option to my program, I came across a problem. I entered an order with two of the same pizza, except one had stuffed crust and one did not. However, it repeated the first order of this pizza twice. The second screenshot is the code for the order output before I fixed this problem. To fix it, I changed the “for pizza in order” to “for pizza in range(len(order))”. So, while originally pizza would be the id of the pizza, now pizza was the position of the id of the pizza in this list. This helped me get around duplicates of the same id in the list, as their pizza value will be different. The third screenshot is the code after I made the fix explained above, and the final screenshot is the same order but with the improved code. |  |
| In version five, I added the option for users to add stuffed crust to any of their pizzas. After every pizza, they are prompted as to whether or not they want to add stuffed crust to that pizza for the price of 3 dollars. If they answer yes, this price is added to the price variable. Regardless of what they answer, their answer gets appended to a list that keeps track of whether or not each pizza has stuffed crust or not. This is included in the output of the user’s order.  These screenshots provided are of various testing examples of this version. There is not a screenshot of the code because that was included in the previous log entry. |  |
| Upon starting version six, I realised that the price for the stuffed crust wasn’t actually adding on to the total price amount. The kiwi pizza is on the regular menu, meaning with a normal crust it costs $8. However, the order in the first screenshot says that the price for a kiwi pizza WITH stuffed crust is $8. This was a very quick fix, I just needed to make the stuffed crust function return the price variable so that the changes I made to the variable are saved. The second screenshot shows the changes I made to the code, and the third screenshot shows the remedied price. |  |
| In version six, I added the option of either delivery or pickup for the order. I put this in a function that gets called at the very beginning of the program. If the user enters D for delivery an extra eight dollars gets added onto the total price. Otherwise, if they enter P, the price is not affected. The third screenshot shows how the delivery address input works, and how the delivery cost gets added to the price. The next two screenshots show the same order, one with delivery and one with pickup. The difference between the prices is eight dollars – the delivery cost. |  |
| Here is how my trello board is looking after completing version six. I think that it is a good amount of progress for the amount of time I still have left. Version six was only just over half of the Other Input card (see the next screenshot), so I have not been able to mark that as done yet. Because the delivery/pickup option was fairly complex, I decided to make it its own version. Finally, the third screenshot is of the calendar, which shows that I am a few days ahead of where I had planned to be. This is good, and means that I am making good progress. |  |
| In version seven I added the name input and a confirmation process at the end of the program. The name input was very simple – just a basic input. However, the confirmation was a bit more complex. Once the user has quit the ordering process, their order is printed out and they have the option to re-do their order as many times as they’d like until they’re satisfied with what they’re ordering. Once they’ve confirmed their order is correct, the process is the same with the delivery details. Their delivery details (or just their name and pickup if they’re picking up their order) are printed out and they confirm whether it is correct. This is especially important because you don’t want the pizza being delivered to the wrong place. The user has the option to just change the details like name and address, or they can change from delivery to pickup (and vice-versa) and the price will fix itself accordingly. This was fairly challenging, as I had to put in a lot of parameters – not just in the confirmation function but in a lot of other functions – so that I could call the necessary functions and edit the variables and lists where necessary.  Shown in the screenshots is two tests where a pizza order is placed, and then details are changed later in the confirmation process. |  |
| In version eight I added error handling for the two sets boundaries in the program – the amount of pizzas that the program is ordering, and the ID of the pizzas. If they try to order more than 3 pizzas the program will stop them and say that it is due to the pandemic. On the other end, if they try to order no pizzas the program will tell them that they have to order at least one pizza. If they try to order a pizza that doesn’t exist the program will ask them to enter a valid pizza ID and send them back to the menu input.  I used a constant called PIZZA\_RESTRICTION because later on the pizzeria may decide that they can take up to five pizzas, for example.  The next three screenshots show testing of the maximum numbers of pizzas that can be ordered. As shown, it works both for the first order and any fix-up orders that the user performs after their initial order. The screenshot after that shows testing of the minimum amount of pizzas.  The next two screenshots show testing of the pizza ID boundaries, making sure the user doesn’t try and order a pizza that isn’t offered.  The final screenshot shows a mixture of this testing, both ID and pizza amount boundaries to make sure they are working together. |  |
| There are a lot of code screenshots for this version, as there were a lot of changes made throughout the program.  In version nine I added error handling for any exceptional unexpected input that the user might enter. This basically means making the program unbreakable so that the user can’t accidentally (or on purpose) crash it. For the delivery/pickup input and the pizza ordering menu input I just added an extra branch to their if/elif/else statements to cater for any unexpected input. For the yes/no input (used in the stuffed crust input and in confirmation at the end of the program) I created a function to replace input that makes sure the user enters valid input. Finally, for the input of the pizza ID, I had to make sure that they entered a whole number otherwise the program would crash. I created another function with a try/except so that if there was a ValueError they would receive an error message and be prompted to try again.  The screenshots after the code show a testing run-through that checks every exceptional input possible. At every possible opportunity I tried to break the program. |  |
|  |  |