Code Evolution Analysis

In this assignment, I created two versions of the same Shopping List Manager program: one “messy” version and one “clean” version. Both versions have the same functionality, but the way the code is written is very different. Here are the main differences and what I learned:

1. Variable Names - Messy Version: I used very short and unclear names like x, y, z, or n. - Clean Version: I used clear names like item1, item2, item3, and item_count. - Why Clean is Better: Good names make the program easier to read and understand. If another person looks at the code, they can quickly see what each variable is for.

2. User Prompts and Messages - Messy Version: Prompts were short and sometimes confusing, like "opt:" or "item:". - Clean Version: I used full sentences like "Choose option:" or "Enter item:". - Why Clean is Better: Clear prompts make the program easier for the user. It reduces mistakes and makes the program look more professional.

3. Formatting and Indentation - Messy Version: The code is squeezed together, with little spacing and sometimes hard to follow. - Clean Version: I used proper spacing, blank lines, and consistent indentation. - Why Clean is Better: Good formatting makes the code more readable. It is easier to find errors and follow the flow of the program.

4. Structure of Code - Messy Version: The logic is written quickly with lots of repeated code. It works but is harder to fix or change later. - Clean Version: The logic is organised. Similar tasks are grouped neatly, and there are helpful comments. - Why Clean is Better: A structured program is easier to maintain. If I want to add more features later, it will be simpler.

5. Comments and Explanations - Messy Version: No comments. Someone else reading the code would have to figure it out on their own. - Clean Version: Added small comments to explain important parts of the code, like when items are added or removed. - Why Clean is Better: Comments help others (and my future self) to understand the purpose of each part of the code.

Reflection The clean version was much easier to understand when I came back to it later. The messy version worked but felt confusing even to me after some time. From this exercise, I learned that good coding practices like clear names, proper formatting, comments, and user-friendly messages are just as important as making the program work.