2-1 Journal: What Makes a Productive Code Review?

Stephen Zaferopolos

Southern New Hampshire University

CS499 Computer Science Capstone

07/13/25

Code review is the use of positive scrutinization of another developer’s code with the intent of identifying errors, inconsistencies in coding practices, or opportunities for improvement. It is important for computer science professionals to participate in code reviews for several reasons. The main reason to code review is to find ways to optimize a piece of code. Whether written by a seasoned professional or a new developer, there are always ways to improve code. Improvements could be as simple as better notations or use of white space to make code easier to read, or as complex as a new method or function that may be more efficient than the current implementation. Code reviews provide a new way of looking at the implementation and choosing the best course of action, whether that be the new suggestion, or the original solution. The second reason to practice code review is to improve our skills as programmers. The original developer can improve their skills by receiving the scrutiny of their work in the spirit it was given and discover a whole new approach to a problem that may not have been obvious without a second viewpoint. The code reviewer could also benefit from reviewing others’ work. The reviewer could learn new ways of solving a problem just as much as the original developer, however the reviewer also can refine their analyzing skills by giving a careful read of another’s work, they could take that skill and apply it to their own work, creating a more efficient product before submitting their own work for code review.

The code reviews the best practices I find crucial pertain to the efficiency and manageability aspects. I find it important to concentrate on a specific portion of code rather than an entire program. Choose a specific method or several methods that work together and concentrate on how that aspect can be improved. A well-written method can be integrated into just about any application. Second would be to utilize a checklist. Establishing expectations before a code review keeps discussions focused and avoids time-wasting tangents. The third most important practice is to foster a positive code review culture. It is important to recognize the difference between positive scrutiny and negative criticism. We are not out to tear down other developers, rather our intent should be to enhance the skills of each other

The software I will be submitting for code review will be the linked list exercise from CS260 “Data Structures and Algorithms”. The code can be found in module three of the course “lab3-2”. This artifact will be the basis for all three of the outcomes for the enhancement portion of the course. I intend to create an outline for the code review of this artifact. I will take the checklist and use it to answer each item specifically , which I will use to make a dry run, improving the outline each time. I tend to struggle with jargon and vocabulary more than implementing. I’d rather do than explain. Much of my outline will revolve around cues for proper jargon, in addition to cues for the points I’d like to make for each item in the checklist. There will only be one artifact to code review, so much of the actual review will be at the beginning of the exercise. As I move from category to category, I wll do a quick review of the artifact, what the last enhancement was, and how the next enhancement will build on the previous. My first enhancement will be to translate the code from C++ to python, the second will be to create a tool within the application to convert the unsecure data from a plaintext csv file into a much more secure MongoDB database. The third outcome will be to transform the tools from the original application from manipulating a linked list, to managing a MongoDB database.