Pushpa Laxman

Southern New Hampshire University

CS-499-19649-M01 Computer Science Capstone 2025

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

Neil Kalinowski  
05/16/2025

Q. Part 1:

Q. What is code review?

The term "code review" refers to exactly what it implies: checking the code again. It is commonly described as a peer review of code, which helps developers ensure or enhance code quality and identify and correct errors that may have been overlooked during the initial development phase before merging and deploying it. In other terms, "Code review is a process of ensuring software quality, during which one or several individuals or tools methodically inspect and assess the program's source code."(SonarSource, 2024). Once a software developer finishes coding, they can easily make mistakes. I have overlooked significant elements in my personal projects and only realized them while testing or debugging. It is inevitable that this will eventually occur. Keeping this in mind, code reviews serve as an excellent opportunity to gain a second perspective on the solutions and implementations, aiding in identifying bugs, logical errors, uncovered edge cases, or other issues so we may solve them collaboratively and avoid mistakes.

Q. Why is it an important practice for computer science professionals?

Code review is a crucial practice for computer science professionals. The goal of this process is to reduce the number of mistakes and their potential impact. In other words, it evaluates and scrutinizes the code written by developers to verify its quality, effectiveness, and adherence to established standards. A review by peers can assist in identifying bugs and issues that the original programmer might have overlooked. By utilizing the skills and insights of several members within the development team, code reviews foster knowledge sharing, the adoption of best practices, and ongoing learning, ultimately resulting in more resilient and dependable software solutions. Moreover, it enhances code quality, encourages knowledge sharing, and promotes collaboration among team members.

Q. What are some code review best practices that you read about in the resources that are crucial to include in a code review? Include when a code review should occur in the development process with a rationale as to why.

Collaborating and gaining insights from others is one of the best aspects of becoming a developer or exploring programming on your own. Code review is a frequent activity, and the conversations that emerge from it will enhance your skills as a programmer, regardless of whether you're assessing your own code or someone else's. In essence, it’s a collective effort.

Reviewing code is an essential part of the software development workflow once a developer completes their coding tasks, Determining the right time to conduct code reviews is a challenging issue, and each development team must establish its own strategy. Some teams review each modification integrated into the main branch, while others do not mandate a review for what they deem minor changes. In this scenario, engineers must weigh the need to conserve time against the importance of upholding code quality.

A code review is recommended before the code is integrated into the main codebase. It should happen after automated checks (tests, style, other CI) have been completed successfully, but before the code merges to the repository’s mainline branch.  It is preferable for developers to detect bugs before customers encounter them, rather than finding out post-launch and rushing to create a fix.

Here are some code review best practices:

1. Develop a checklist for code reviews containing a fixed set of questions and guidelines that the reviewing team will adhere to throughout the code evaluation process before granting permission for code integration into the codebase. This may encompass aspects such as Readability, Security, Test Coverage, Architecture, Reusability, etc.

2. Implementing code review metrics can assist you in evaluating the effectiveness of reviews, assessing how changes affect the process, and estimating the hours required to finish a project.

3. Code should be tested before it is submitted for review, so that its functionality can be confirmed before asking for feedback.

4. Code reviews are notorious for taking too long, so adhering to best practices by automating as much as feasible is advisable.

In addition to that, I think it's important for developers to conduct a code review right after writing the code. Since the code can be quite detailed, reviewing it effectively with multiple changes addressing various objectives becomes increasingly challenging. The best practice in code review proves beneficial when utilizing code revision tools like git. The GitHub platform serves as an excellent resource for developers. They can upload their code for team evaluation, identify any mistakes, and rectify them immediately. After resolving the issues, they can commit the updates and prepare the revised code for use. Performing unit tests for each small change would be a recommended practice to verify the code's functionality and consequently maintain compliance with the coding standards.

Part 2:

Q. What software have you chosen to use to record your code review?

I utilized ScreenPal to capture the code review. The ScreenPal application has a recording limit of 15 minutes. Consequently, I have uploaded three videos corresponding to each article category: Software Engineering and Design, Algorithms and Data Structures, and Database.

Q. Describe your approach to creating an outline or writing a script for your code review for each of the three categories that you will be reviewing based on the rubric as well as the code review checklist.

Code review videos feature an assessment of the three chosen artifact codes for the capstone project, focusing on their shortcomings and vulnerabilities, along with a description of my strategy for improvements.

The code review involves a detailed examination of the current source code, providing a concise overview of the project, its purpose, and what the application accomplishes. Following this, I assessed the provided checklist in relation to the existing source code, discussing its limitations and sharing my insights on the proposed enhancements across three categories: software design and engineering artifacts, algorithms and data structures, and databases.

Through the creation of these videos, I am enhancing my skills in professional communication across written, verbal, and visual channels. Furthermore, I am ensuring that my code review presentation is effectively communicated in a collaborative setting and expressing my ideas clearly.

In the artifact of the **Code analysis**, I illustrate the key components of: Current code and its operation: This segment examines the capabilities and performance of the current code.

**Code Review**: Highlight areas requiring improvement concerning structure, documentation, security, testing, loops, functionality, comments, etc.

**Planned Improvements**: This portion outlines intended enhancements that respond to the concerns identified in the code review.

**Software Design and Engineering Enhancements**

The chosen artifact for the code review of software design and engineering is the Weight Tracking Mobile App. This application was created as part of the computer science course CS360 Mobile Architecture and Programming. Its primary objective is to allow users to monitor their weight through an Android device. The mobile app was developed in JAVA and incorporates an SQLite database. Planned upgrades include implementing input validation for usernames and passwords, improving error handling, creating a new account page for new users, and adding graphs or charts to assist users in tracking their weight progress or changes. With these enhancements, I aim to improve data security, enhance account management and login functionalities, safeguard account information, ensure accessibility, and make the interface more user-friendly while also addressing exception handling.

**Algorithms and Data Structure Enhancements**

The code review for algorithms and data structures pertains to the Animal Shelter class. This program was planned, designed, and developed as part of the computer science curriculum, CS 340 Client-Server Development course. It aims to create a fully functional interactive web-based dashboard for the Grazioso animals training company, which specializes in identifying suitable dogs for their search and rescue training programs. The application is built using Python and the Dash framework, with MongoDB acting as its database. The connection between Python and MongoDB is established using the PyMongo driver, while Jupyter Notebook functions as a testing environment for the web application. The application is structured as a monolithic architecture, which limits its flexibility and scalability due to a tightly coupled codebase. I intend to decompose the code into smaller, individual services, ensuring that each service can be independently built, developed, and tested.

Additionally, I plan to utilize advanced operations, as the list will facilitate sorting algorithms like quicksort and binary search, which are beneficial for dynamic data in this complex application. With these enhancements, I aim to improve the data structures and algorithms of the Animal Shelter class, thereby boosting the overall performance and value of the system.

**Databases Enhancements**

Code reviews in databases are an artifact of the Grazioso Salvare Search for Rescue Web Application. This web application was developed as part of the CS340 Client-Server Development course. Leveraging a pre-existing database from animal shelters, the application identifies and classifies available dogs for training in various rescue programs by developing a web interface. The project utilizes MongoDB alongside the Dash framework within the Python application. Integration between MongoDB and Python is achieved through the PyMongo library. Assessments of the web application are performed using Jupyter Notebook. The planned enhancement is to utilize advanced features of MongoDB by setting up indexes on commonly queried attributes such as age, animal ID, type, and color. This will allow the database to efficiently find data quickly instead of scanning the whole table, facilitating considerable horizontal scalability without necessitating changes to the application logic. The data entered the database is not validated, making it vulnerable to security risks, so I plan to implement strict input validation as my second enhancement to the project. In addition, I intend to execute the operations entirely within the database using Aggregation pipelines, eliminating the need for extensive result sets to be transferred over the network. The third improvement I have made to the project is this one.

**References**

*Choosing the best code review method: 4 types explained. Nulab. (n.d.). https://nulab.com/learn/software-development/code-reviews/*

*Ghani, U. (2023, September 18). 5 code review best practices. Work Life by Atlassian. https://www.atlassian.com/blog/add-ons/code-review-best-practices*

*Greiler, Dr. M. (2023, January 15). 30 Proven Code Review Best Practices from Microsoft - dr. McKayla. Michaela Greiler. https://www.michaelagreiler.com/code-review-best-practices/*

*Lazar, M. (2024, September 20). Why code reviews are essential for software projects’ success. HyperSense Blog. https://hypersense-software.com/blog/2024/08/14/importance-of-code-reviews-in-software-development/*

*Rokem, A. (2024, September 5). Ten simple rules for scientific code review. PLoS computational biology. https://pmc.ncbi.nlm.nih.gov/articles/PMC11376551/*

*SonarSource. (2024, April 11). What is a code review?: Definition & Guide. Sonar. https://www.sonarsource.com/learn/code-review/*