Module Two Journal

CS – 499: Computer Science Capstone

Benjamin Leanna

SNHU

2024 – 03 – 17

1. What is code review?

Code review is a methodical examination of source code aimed at identifying and rectifying defects, ensuring adherence to coding standards, and enhancing overall code quality. This collaborative process involves peers or team members scrutinizing code changes before they are merged into the project's codebase.

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

For computer science professionals, code review holds significant importance for various reasons. Firstly, it plays a crucial role in maintaining code quality by uncovering bugs, potential security vulnerabilities, and architectural flaws early in the development cycle, thereby minimizing the risk of costly errors down the line. Additionally, code review fosters knowledge sharing and collaboration within teams, allowing developers to gain insights into diverse approaches and solutions. Moreover, it serves as a vital quality assurance measure, ensuring that the software meets specified requirements and functions as intended.

1. 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.

Implementing certain best practices can enhance the effectiveness of code reviews. It's advisable to conduct reviews regularly throughout the development process, rather than waiting until the end, to catch issues early and provide timely feedback. Breaking down reviews into smaller, manageable chunks ensures thoroughness and efficiency. Maintaining a constructive and respectful tone during reviews is essential, focusing on evaluating the code rather than criticizing the developer personally. Also, incorporating automated tools for code analysis and utilizing checklists can streamline the review process and ensure consistency. You shouldn’t review code too quicky, but at the same time, the review should not be too long either; preferably no more than 60 minutes at a time.

In my thoughts, code reviews should occur at various stages of the development process, including before committing code changes to version control, before merging branches, and during code refactoring. Conducting reviews at these junctures helps detect issues early, minimizing their impact on project timelines and reducing the risk of introducing regressions.

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

As for recording my code review, I will be using my built-in Alienware screen capture software. I will then find a place to host this video depending on the size constraints of the video.

1. 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 rebric as well as the code review checklist.

Approach to Creating an Outline for Code Review:

Category One: Software Engineering and Design:

Adherence to Coding Standards:

Review the codebase to ensure consistency in coding style, naming conventions, and formatting. (Addresses: Design, develop, and deliver professional-quality oral, written, and visual communications)

Check for adherence to established design patterns and architectural principles. (Addresses: Design and evaluate computing solutions)

Evaluate the modularity and scalability of the codebase. (Addresses: Employ strategies for building collaborative environments)

Functionality and Logic:

Test the functionality of implemented features against specified requirements. (Addresses: Design and evaluate computing solutions)

Identify any logical errors or inconsistencies in the code. (Addresses: Develop a security mindset)

Review error handling mechanisms and edge case scenarios. (Addresses: Employ strategies for building collaborative environments)

Performance and Optimization:

Analyze the efficiency of algorithms and data structures used in the code. (Addresses: Demonstrate an ability to use well-founded and innovative techniques)

Evaluate the performance impact of resource-intensive operations. (Addresses: Develop a security mindset)

Look for opportunities to optimize code for speed, memory usage, and scalability. (Addresses: Design and evaluate computing solutions)

Category Two: Algorithms and Data Structures:

Sorting and Searching Algorithms:

Evaluate the implementation and efficiency of sorting and searching algorithms. (Addresses: Demonstrate an ability to use well-founded and innovative techniques)

Assess the suitability of chosen algorithms for the specific requirements of the project. (Addresses: Design and evaluate computing solutions)

Identify opportunities for optimization to improve algorithmic performance. (Addresses: Develop a security mindset)

Data Structure Usage:

Review the selection and implementation of data structures such as arrays, linked lists, and trees. (Addresses: Design and evaluate computing solutions)

Evaluate the appropriateness of data structures for storing and manipulating data. (Addresses: Design, develop, and deliver professional-quality oral, written, and visual communications)

Assess the efficiency and scalability of data structures in handling expected workloads. (Addresses: Employ strategies for building collaborative environments)

Category Three: Databases:

Database Integration:

Verify the integration of database operations within the codebase. (Addresses: Design and evaluate computing solutions)

Review SQL queries and database interactions for correctness and efficiency. (Addresses: Demonstrate an ability to use well-founded and innovative techniques)

Ensure proper error handling and data validation mechanisms are in place. (Addresses: Develop a security mindset)

Data Management and Security:

Assess the handling of sensitive data and adherence to privacy regulations. (Addresses: Develop a security mindset)

Review database schema design for normalization and optimization. (Addresses: Design and evaluate computing solutions)

Evaluate mechanisms for data backup, recovery, and access control. (Addresses: Employ strategies for building collaborative environments)

References:

*Best practices for code review*. (n.d.). smartbear.com. https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/

*Software Engineering Best Practices for Code Review*. (2024, March 15). https://daily.dev/blog/software-engineering-best-practices-for-code-review