**CS 499**

**2-1 Journal**

Matthew Berdecia

Southern New Hampshire University

**Part 1: Understanding Code Review**

Code review is a systematic examination of source code aimed at identifying and fixing mistakes that may have been overlooked during the initial development phase. It improves the overall quality of the software by ensuring it adheres to coding standards, is free from vulnerabilities, and performs efficiently (McConnell, 2004). This practice is crucial for computer science professionals as it fosters a culture of collaboration and continuous learning within development teams (Rigby & Storey, 2011).

Best practices for code review include setting clear goals, using a checklist for consistency, focusing on the code rather than the coder, and limiting review sessions to manageable chunks to maintain focus and effectiveness (Bacchelli & Bird, 2013). Code reviews should occur after significant development phases or before merging new features into the main branch to catch issues early and reduce the cost of fixing bugs (Fagan, 1976).

**Part 2: Software and Approach for Code Review**

For recording my code review, I have chosen GitHub, as it integrates seamlessly with version control and provides robust tools for commenting, suggesting changes, and tracking the review process (Loeliger & McCullough, 2012).

My approach to outlining or scripting the code review involves focusing on three key categories: functionality, readability, and maintainability. In the functionality category, I ensure the code meets all specified requirements and passes the necessary tests. For readability, I check that the code is well-commented, adheres to coding standards, and is easy to understand. In maintainability, I evaluate the code's structure, ensuring it is modular and can be easily modified with minimal technical debt (Martin, 2008). This structured approach aims to provide constructive feedback that enhances the quality and maintainability of the codebase.

**References**

Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. *Proceedings of the 2013 International Conference on Software Engineering*.

Fagan, M. E. (1976). Design and code inspections to reduce errors in program development. *IBM Systems Journal*, *15*(3), 182-211.

Loeliger, J., & McCullough, M. (2012). *Version control with Git: Powerful tools and techniques for collaborative software development*. O'Reilly Media, Inc.

Martin, R. C. (2008). *Clean code: A handbook of agile software craftsmanship*. Prentice Hall.

McConnell, S. (2004). *Code complete: A practical handbook of software construction*. Microsoft Press.

Rigby, P. C., & Storey, M. A. (2011). Understanding broadcast-based peer review on open source software projects. *Proceedings of the 33rd International Conference on Software Engineering*.