### **Part 1: Code Review Overview**

Code review is the systematic evaluation of computer source code to identify errors, ensure adherence to coding standards, and assess overall quality. It is a critical practice in software development for detecting bugs and enhancing maintainability. According to the OWASP Code Review Guide, this process plays a pivotal role in identifying vulnerabilities early in the software development lifecycle, ensuring that applications are secure and robust before deployment. The importance of code review lies in its ability to detect defects early, saving time and costs associated with fixing issues later in production. Code reviews also promote knowledge sharing and collaboration, allowing team members to learn from one another while establishing a culture of quality and security.

Best practices in code review include defining the review's scope to focus on critical modules or areas prone to errors or vulnerabilities, utilizing structured checklists for consistency, and fostering a collaborative environment that encourages learning and constructive feedback. Tools like static analysis can complement manual reviews by identifying potential issues, while human reviewers provide the necessary context and risk assessment. Code reviews should occur throughout the development process, particularly during pre-commit, post-commit, and periodic audits. Pre-commit reviews ensure to reduce errors that enter the codebase, while post-commit reviews catch integration issues, though they may require rework. Periodic audits target specific modules or vulnerabilities, as well as ensuring compliance for various security frameworks. Conducting reviews at all of these stages balances efficiency with quality, and supports immediate needs while ensuring long-term reliability.

### **Part 2: Software and Approach**

For this code review, I have chosen my code which I have chosen to enhance for this capstone course. I have chosen this so that I have a framework to improve my code from, and also gain insight into my work from a fresh perspective since some time has passed from the project’s last commit.

For software design and engineering, I will review the existing code's functionality by analyzing its features and operations. The goal is to clearly describe what the code does, including its structure, logic, and functionality. Using the checklist, I will evaluate the design for weaknesses such as deviations from secure coding practices, inefficient logic, or insufficient error handling. Specific criteria will include whether the code adheres to coding standards, uses modular design, and maintains proper commenting and documentation.

For algorithms and data structures, I will focus on the efficiency and functionality of the algorithms and data structures used in the code. I will describe the existing code's purpose, detailing its computational processes and how data is managed. Using the checklist, I will assess the code for efficiency, ensuring that operations are optimized and unnecessary redundancies are minimized. For example, I will check for inappropriate use of loops or inefficient sorting techniques.

The database review will involve analyzing how the code interacts with data storage systems, focusing on functionality, security, and efficiency. I will describe the code’s handling of database operations, such as querying, updating, and managing data. Using the checklist, I will critically evaluate database interactions, highlighting weaknesses like poor query optimization, lack of parameterized queries, or inadequate handling of sensitive data.

For each category, I will use the CS 499 checklist as a foundation to systematically evaluate the code. Any issues pertaining to the three categories will be acknowledged and noted for improvement.