CS-499 Module Two Journal

Darrell Lindsey

09/12/2025

Code review is a process where developers take time to examine and evaluate code written by themselves or others to ensure it meets quality standards. It’s not just about finding bugs, it’s about improving the overall structure, readability, security, and maintainability of the code. For someone like me, transitioning from a hardware and manufacturing background to software development, code review is a great way to sharpen my skills and learn from different perspectives. It forces you to slow down and think critically about how your code works, how it could break, and how it could be improved.

In the professional world, code review is essential. It helps teams catch issues early, share knowledge, and maintain consistency across a project. It also encourages better coding habits and reinforces accountability. When done right, it can improve performance, reduce technical debt, and make future updates easier. From what I’ve read in the CS 499 resources, some of the most important best practices include checking for secure coding (like validating input and avoiding hardcoded values), making sure error handling is in place, and reviewing for clarity and modularity. Code should be easy to read and well-documented so that anyone on the team can understand it. Performance and scalability are also key, especially when working with algorithms or databases.

Code reviews should happen after the initial development phase but before merging code into the main branch. This timing allows developers to catch problems before they affect the rest of the system. It also gives the author a chance to make improvements while the code is still fresh in their mind. In agile environments, code reviews are often built into the workflow through pull requests, which makes them a regular part of the development cycle.

For my code review video, I’ll be using PowerPoint’s screen recording feature to walk through my code directly. I won’t be using slides, I’ll be recording my screen as I navigate through the actual codebase, pointing out key sections and narrating my analysis and enhancement plan. This approach allows me to focus on the code itself, which I think is the most effective way to demonstrate both the current functionality and the improvements I intend to make. I’ve practiced using PowerPoint’s recording tools so I feel confident about capturing both the visuals and my narration clearly.

To prepare, I created a structured outline based on the CS 499 rubric and the code review checklist. I broke the review into three categories: software engineering and design, algorithms and data structures, and databases. For each category, I started by identifying the core functionality of the code, what it does, how it’s structured, and how it supports the overall goals of the application. Then I used the checklist to analyze the code critically, looking for things like modularity, error handling, secure coding practices, performance, and documentation. I made notes on areas that could be improved, such as better separation of concerns, more efficient algorithms, and stronger database security. Finally, I outlined a clear plan for each category, making sure to tie those improvements directly to the CS 499 course outcomes.

For the software engineering section, I focused on the modular design of my Inventory Management App and identified areas where I could improve accessibility and documentation. In the algorithms section, I looked at how I handled sorting and searching and saw opportunities to optimize performance and introduce better data abstraction. For the database section, I reviewed how I handled CRUD operations and found ways to improve query safety, input validation, and schema design. This approach helps me stay focused during the recording and ensures that I cover all the required elements in a logical, professional way.