1. What is code review?

Code review is when developers look over each other’s code to catch mistakes, improve quality, and make sure it follows project standards. It helps ensure the code is clear, functional, and easy to maintain. Code reviews also give developers a chance to learn from one another and keep the whole team on the same page (SmartBear, n.d.).

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

Code review is important because it helps catch bugs early, improves security, and keeps code consistent. It’s also a great way for developers to learn from each other and share new techniques. According to OWASP (2017), code reviews can catch security issues before software goes live, making the product safer and reducing long-term risks.

3. What are some code review best practices that are crucial to include in a code review? Include when a code review should occur in the development process with a rationale.

Some best practices include reviewing small chunks of code (under 400 lines), using checklists, and taking breaks to stay focused (SmartBear, n.d.). Code reviews should happen before the code is committed, so any problems can be fixed early. This helps avoid bugs or security issues from reaching later stages of development or production (OWASP, 2017).

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

For my code review, I’ll use OBS Studio. It’s a free screen recording tool that lets me walk through the code while explaining what I’m reviewing. It’s useful for capturing both the code and my commentary in one video, which makes the review easier to follow and share.

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

To stay organized, I’ll structure my code review into three parts based on the CS 499 checklist: structure, documentation, and logic. I’ll first look at whether the code follows the design and avoids repeated or unused code. Then I’ll check if comments and variable names are clear and consistent. Lastly, I’ll review logic and defensive programming, making sure loops, inputs, and edge cases are handled properly (George, 2025).

References:

George, C. (2025). CS 499 transcript for checklist for code reviews [Transcript]. Southern New Hampshire University.

OWASP Foundation. (2017). OWASP code review guide v2.0. https://owasp.org/www-project-code-review-guide/assets/OWASP\_Code\_Review\_Guide\_v2.pdf

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