**Part 1**

**What is code review?**  
Code review is when another person looks over your code to check if it works the way it should and follows good practices. It’s like proofreading an essay, but for programming. The goal is to find mistakes, suggest improvements, and make sure the code is easy for others to read and use.

**Why is it an important practice for computer science professionals?**  
Code review is important because it helps catch errors before they become bigger problems. It also makes the code more consistent across a project, which makes it easier for teams to work together. On top of that, it’s a way for developers to learn from each other and share knowledge, which makes everyone better over time.

**What are some code review best practices that you read about in the resources that are crucial to include in a code review?**

Keeping reviews short and focused so they don’t feel overwhelming. Giving feedback in a helpful way, instead of just pointing out what’s wrong. Using a checklist to make sure things like readability, testing, and security are all covered. Using tools to handle basic things like formatting, so reviewers can focus on the logic and design of the code.

**When should a code review occur in the development process, and why?**

A code review should occur before code is merged into the main branch of the repository. Doing it early stops bugs and bad habits from spreading into the larger codebase, which saves time later on. It’s easier to fix problems before they become part of the final product.

**Part 2**

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

I plan to use Microsoft Word to prepare my script and outline. This helps me organize my thoughts and make sure I cover everything. If I need to, I can also use it to add screenshots of the code so my review is clear and easy to follow.

**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 rubric as well as the code review checklist.**

My approach is to first review the rubric and checklist so that I know the exact categories to address. To stay organized I’ll break my review into three main categories based on the rubric and checklist:

Functionality – I will describe if the code works the way it should. In my script I’ll write about how the program runs and whether it works correctly or not.

Readability and Style – I’ll discuss at how easy the code is to read and understand to people that haven’t seen the Travlr Getaways project. I’ll check if the names of variables and functions make sense, if the formatting is the same throughout the script, and the comments.

Design and Maintainability – This is where I need to mention the organization and the way it is built. It should be built so it is easy to update in the future. I’ll also check if my project has good structure and doesn’t repeat the same code in different places.

**References**

Bacchelli, A., & Bird, C. (2013). Expectations, outcomes, and challenges of modern code review. *Proceedings of the 2013 International Conference on Software Engineering*, 712–721. https://doi.org/10.1109/ICSE.2013.6606617

Microsoft. (2022). *Best practices for code reviews*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests-code-review

Rigby, P. C., & Bird, C. (2013). Convergent contemporary software peer review practices. *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, 202–212. https://doi.org/10.1145/2491411.2491444

SmartBear. (2023). *Best practices for code review*. SmartBear Software. https://smartbear.com/learn/code-review/best-practices-for-peer-code-review/