**CS 405 Journal: Portfolio Reflection**

Throughout this course, I gained a strong understanding of why security should be integrated early in the software development process. One of the most important lessons was adopting secure coding standards from the beginning. Waiting until the end to address security makes it harder to fix issues and increases overall costs. By following the SEI CERT C++ Coding Standard, I learned how to avoid unsafe functions, reduce undefined behavior, and write more secure and maintainable code. Practices such as input validation, limiting pointer use, and applying constant expressions helped me see how structured coding reduces the risk of vulnerabilities.

Another key focus of this course was evaluating and managing risk. Not all threats are equal, so learning how to assess both the likelihood and impact of vulnerabilities helped me make informed decisions. Cost-benefit analysis plays an important role when deciding whether to fix, mitigate, or accept a risk. This approach was especially useful when selecting coding standards and security tools. By documenting examples and explaining why each rule mattered, I better understood how to prioritize actions and resources effectively.

The concept of zero trust changed how I think about security. Instead of assuming anything inside a system is safe, the zero trust model requires verification at every level. This includes applying least-privilege access, enforcing strong authentication, and continuously monitoring activity. I included these principles when writing the security policy for Green Pace, which emphasized access control, auditing, and secure data handling. This approach is essential in modern environments where networks are often distributed and remote access is common.

Writing and recommending security policies helped me connect theory to real-world practices. The authentication, authorization, and accounting policy I created ensures proper user identity verification, permission management, and activity tracking. I also developed policies around encryption and coding standards, and integrated automated tools such as Cppcheck to help catch issues during development. These policies support a DevSecOps pipeline that includes security at each phase, from code writing to deployment.

This course taught me to treat security as a core part of software development rather than an afterthought. Whether through secure coding, risk evaluation, trust management, or policy creation, I now understand how to build systems that are both functional and protected. I plan to include this project in my portfolio to highlight my growth in secure software engineering and my ability to apply these principles in real scenarios.