**Portfolio Reflection – CS 405**

**Jose Munoz**

**Adoption of Secure Coding Standards**

Throughout this course, I’ve learned that security cannot be an afterthought. Waiting until the end of development to address vulnerabilities creates expensive and dangerous gaps. Adopting a secure coding standard early ensures that every step of the software development life cycle integrates security. For example, consistently validating inputs and managing memory prevents common issues like buffer overflows or injection attacks. These lessons connect back to earlier coding activities where proper range checking and parameter validation were required. Embedding these practices from the start builds software that is secure by design rather than patched later.

**Risk Evaluation and Cost-Benefit of Mitigation**

Evaluating and mitigating risks involves balancing resources with the impact of potential threats. One key takeaway from this course is that not every risk has the same weight. Performing cost-benefit analyses helps prioritize which threats to mitigate immediately versus those that can be addressed later with monitoring and compensating controls. For example, encrypting all sensitive user data has an upfront cost but prevents much higher costs associated with breaches. The Sony Pictures breach and others we studied showed how underestimating risks can lead to devastating financial and reputational losses.

**Zero Trust Security**

Zero Trust was another major concept emphasized throughout the course. It operates on the idea of “never trust, always verify.” Instead of assuming that internal networks are safe, Zero Trust enforces continuous authentication and strict access control. This model aligns closely with secure coding practices by requiring principle-of-least-privilege designs in applications. For example, a module should only have the minimal access it needs, reducing the attack surface if an account or system is compromised. Implementing Zero Trust policies has become a best practice across industries, and it’s something I plan to integrate into future projects.

**Implementation and Recommendations for Security Policies**

This course also highlighted the importance of well-defined security policies. Policies should provide clear guidelines for secure development, risk assessments, and incident response. Recommendations for improvement include regular policy audits, employee training, and integration of automated security tools such as static analysis and penetration testing during development. One critical improvement I observed is the need to avoid “security silos.” Policies should ensure cross-team collaboration so that security is everyone’s responsibility, not just the IT team’s.