**CS 499 -- Milestone Two Narrative**

**Briefly describe the artifact. What is it? When was it created?**

My artifact is an Inventory Management App I built for CS 360 earlier this year. It's an Android app in Kotlin that lets users create accounts, log in, add inventory items, and get notifications when stock is low. The app uses SQLite for local storage and includes SMS alerts for low stock items.

**Justify the inclusion of the artifact in your ePortfolio. Why did you select this item? What specific components of the artifact showcase your skills and abilities in software development? How was the artifact improved?**

I picked this project because it was already in Kotlin but had major architectural and security problems I could fix to demonstrate professional development skills.

The biggest enhancement was implementing MVVM (Model-View-ViewModel) patterns. My original code-mixed business logic directly in UI components, which made it messy and hard to maintain. I restructured the app by creating utility classes like SecurityUtils, ValidationUtils, NotificationHelper, and Constants that handle business logic separate from UI components. The UI components now focus only on display while proper state management handles data flow between layers.

For security improvements, I replaced plain text passwords with SHA-256 hashing, added input sanitization to prevent injection attacks, and implemented proper permission handling. I also organized the code better by creating modular utility classes instead of repeating code throughout the app and centralizing all hardcoded strings into Constants.kt.

**Did you meet the course outcomes you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?**

Yes, I met my planned outcomes. The MVVM architecture demonstrates proper separation of concerns and professional code organization for software engineering. Password hashing, input validation, and defensive programming show understanding of security best practices. The modular approach shows I can evaluate existing designs and implement better solutions.

No updates to my plans needed since these enhancements directly address what I originally intended to demonstrate.

**Reflect on the process of enhancing and modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?**

Understanding how to properly separate business logic from UI components was harder than expected. I had to completely rethink how data flows through the app and learn MVVM principles properly. Adding password hashing wasn't just about changing the storage method - I had to handle backward compatibility for existing users and implement proper verification systems. Refactoring existing code to follow MVVM principles taught me why planning architecture from the beginning is so important.

The biggest challenges were breaking apart the tangled original code and reorganizing it into proper MVVM layers, which required careful analysis of dependencies. Implementing SHA-256 hashing while maintaining compatibility with existing user accounts was also tricky. Learning how to properly manage data flow between model, view, and viewmodel components took considerable research and testing.

The enhancement tips I learned include starting with a clear architectural plan before coding, implementing security from the beginning rather than retrofitting it, and understanding that modular design makes code much easier to test and maintain. Proper separation of concerns prevents bugs and makes debugging easier.

This process showed me the difference between functional student code and professional-quality software. The MVVM architecture, security improvements, and modular design demonstrate I can build maintainable, secure applications following industry standards.