CS-499 Milestone Four

Darrell Lindsey

10/05/2025

**Database Artifact Enhancement Narrative**

**Artifact Description**  
The artifact I selected for the database category is FirebaseRepository.kt, a Kotlin class responsible for managing real-time data operations in my Inventory Management App. It was originally developed during the early stages of this course as part of my transition from a local SQLite implementation to Firebase Realtime Database. This class encapsulates essential database functionality, including reading, writing, updating, and deleting inventory items. It also leverages Kotlin coroutines to ensure asynchronous, non-blocking execution across the app’s data layer.

**Justification for Inclusion**  
I chose this artifact because it represents a significant evolution in my understanding of cloud-based database architecture and real-time data synchronization. It showcases my ability to design structured data models, implement secure and scalable CRUD operations, and integrate Firebase with Kotlin coroutines for non-blocking execution. Enhancements included adding structured error handling, retry logic for offline resilience, and input validation to prevent malformed data from reaching the database. These improvements demonstrate my growth in designing robust, production-ready database layers.

**Course Outcomes Met**  
This enhancement directly supports the following Computer Science program outcomes:

* Software Engineering/Database: I applied innovative techniques and tools (Firebase, Coroutines, structured models) to implement a solution that delivers real-world value.
* Security Mindset: I added input validation and error handling to mitigate risks such as malformed data, unauthorized access, and sync failures.
* Communication: Through code documentation and structured naming conventions, I ensured the artifact is readable and maintainable for future collaborators.

Although the artifact was originally created in Module Two, the enhancement goal was formally established in Module Five, where I committed to demonstrating mastery in cloud database integration and error resilience. I believe I’ve met that goal. However, I now plan to extend this further by implementing role-based access control and field-level validation to deepen the security layer.

**Reflection on Enhancement Process**  
Enhancing this artifact taught me the importance of designing for failure. Firebase’s real-time sync is powerful, but without proper error handling, it can silently fail or overwrite data. Implementing retry logic and structured exception handling forced me to think critically about edge cases and user behavior. One challenge I faced was managing coroutine scopes across multiple UI components without leaking memory or blocking the main thread. I resolved this by using lifecycle-aware scopes and modularizing the repository interface.

This process also reinforced the value of testability. By injecting dependencies and abstracting Firebase calls, I enabled mocking and unit testing of the repository logic, which will be reflected in future test coverage improvements.