CS-499

3-2 Milestone

Logan Riedell

3/25/2025

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

Originally developed in Java during my previous classes, the artifact is a console-based appointment system based on Java. Using simple service classes and in-memory data, it was meant to control appointments, contacts, and chores. As part of my CS 499 ePortfolio effort, I recently started recreating it in Python.

**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 chose this relic because it is a well-rounded example of a basic software project spanning several domains: object-oriented programming, data validation, service logic, and (lastly) database integration. It provided a clear beginning point for me to show how I may transform a working system into one more modular, scalable, and compliant with professional standards. Separating issues into models, services, and repositories, providing validation through custom exceptions, and planning for SQLite connection for real data persistence has helped the Python rebuild enhance the structure.

**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?**

I have achieved good development toward the intended results. I have certainly worked toward results connected to modular system architecture and using established development approaches. Though everything else is lining up, the database outcome is still under development. Although I have no significant revisions to my result goals right now, I might review them when I incorporate more sophisticated tools and finish SQLite's integration.

**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?**

Rebuilding this Python system has let me see the value of separation of concerns and good architecture. Writing code that works is one thing; another is rewriting it to be maintainable and testable. Designing the repository and service layers in a way that makes it simple to swap out in-memory storage with a true database has proved one of the toughest tasks thus far. I also gained a lot of knowledge about planning; even little choices about file organization or exception definition can significantly affect the project's manageable over-time impact.