Holly Renfrew

Due Date: November 30, 2025

Module 5 Narrative

Artifact Description

The artifact I selected is a console-based text adventure RPG written in Python. I originally created this game in IT-140 as a final project to practice core programming concepts like control flow, functions, conditionals, and user input handling. The game allows the player to move between rooms, fight enemies in turn-based combat, manage weapons and items, and make choices that affect progress through the story.

In the original version, all game data such as player stats, inventory, and room state existed only in memory during a single play session. Once the program was closed, all progress was lost.

Justification for Inclusion in the ePortfolio

I chose this artifact because it shows clear growth from an introductory programming project into something that more closely resembles a real software application. The original game worked, but it did not reflect how data is handled in real-world programs. By enhancing it with database-backed save and load functionality, I was able to improve both its technical quality and its usability.

This artifact highlights my ability to design and integrate a data persistence system using SQLite, organize game state into structured data, and refactor an existing project without breaking its original functionality. Features like save slots, save previews, overwrite warnings, and confirmation prompts demonstrate attention to both software design and user experience. These changes significantly improved the project compared to its original in-memory-only implementation.

Course Outcomes and Planned Coverage

Yes, I met the course outcomes I planned to address in Module One. My original goal for this artifact was to move beyond temporary in-memory storage and implement a real database solution, and that goal was fully achieved.

This enhancement shows progress toward several Computer Science program outcomes. It demonstrates my ability to design a computing solution that solves a practical problem using appropriate tools and data organization. It also shows that I can apply industry-relevant techniques, such as structured state management and database-backed persistence, to improve an existing system in a meaningful way.

I did not need to change my outcome-coverage plan, but the final implementation went further than expected by supporting multiple save slots and user-facing safety checks when saving and loading data.

Reflection on the Enhancement Process

Working on this enhancement taught me a lot about how real applications manage state over time. Adding a database was not just about saving values, but about carefully deciding what needed to be saved, when it needed to happen, and how to reload everything without causing inconsistencies.

One of the biggest challenges was dealing with how interconnected the game systems already were. Because many parts of the game rely on shared global objects, I had to build a clean way to export the current state and reapply it during loading without rewriting the entire codebase. This pushed me to think more carefully about structure and responsibilities between modules. I also ran into issues with imports, game flow, and making sure loading a saved game skipped character creation and story setup. Solving these problems improved my understanding of program control, debugging, and maintaining larger projects.

Overall, this enhancement helped shift my mindset from writing a program just to make it work to designing a system that is maintainable, reusable, and closer to real-world software development.