**Joshua Emmons**

**CS 499 5-2 Databases**

**Milestone Four Narrative – Databases Enhancement**

**Artifact Description**

The selected artifact is a Python-based command-line application titled **Book Collection Tracker**, initially developed during **DAD 220: Introduction to Structured Database Environments**. The purpose of the project was to apply foundational SQL skills in a real-world scenario involving the storage, retrieval, and management of structured data using a relational database.

The original version was designed to demonstrate the creation of an SQLite database containing a single table, books, which stores attributes like title, author, genre, and year. It featured basic functionality such as adding new books, viewing the full list of records, and deleting entries. However, it used raw string interpolation in SQL statements, lacked input validation, and was limited in query flexibility.

**Justification for Inclusion in ePortfolio**

This artifact was chosen for my ePortfolio because it clearly demonstrates growth in database competency, secure coding practices, and software enhancement through iterative development. It reflects my ability to:

* **Design and normalize a database schema**
* **Write and optimize SQL queries**
* **Implement parameterized statements to defend against SQL injection attacks**
* **Build an interactive CRUD interface in Python**
* **Bridge procedural logic with data storage and retrieval systems**

The **enhanced version** of the Book Collection Tracker includes several improvements that align with industry expectations for database-driven applications:

* **Security Enhancements**: Replaced string-based SQL queries with **parameterized queries** using SQLite's ? placeholders to prevent SQL injection.
* **Input Validation**: Added safeguards to ensure the title field is not empty when adding records.
* **Improved Functionality**: Introduced a **search feature** to query records by title using LIKE statements.
* **Modular Code Structure**: Refactored the code to isolate functionality into discrete, reusable methods, improving maintainability and readability.

These improvements demonstrate not only technical skills in SQL and database management but also a strong understanding of software development practices such as iterative improvement, secure coding, and user-focused design.

**Outcome Alignment**

This enhancement aligns with the following **program-level outcome** from the Bachelor of Science in Computer Science curriculum:

*“Demonstrate an ability to use well-founded and innovative techniques, skills, and tools in computing practices for implementing computer solutions that deliver value and accomplish industry-specific goals.”*

The planned enhancements for this artifact, proposed in Module One, were fully achieved. The final implementation reflects intentional efforts to meet objectives in data security, usability, and design effectiveness. Therefore, no updates are necessary to the outcome-coverage plans.

Additionally, this project also supports the development of a **security mindset**, as required by the capstone rubric. The inclusion of **parameterized queries** mitigates SQL injection vulnerabilities—a core learning outcome from both **DAD 220** and later secure coding coursework.

**Enhancement and Learning Reflection**

Enhancing this project provided valuable hands-on experience with real-world considerations for working with structured databases in application development. Through the process, I revisited and reinforced key concepts learned in **DAD 220**, including:

* Relational schema design
* SQL syntax and query formulation
* The importance of **ACID** properties in transaction management
* Defense against common security flaws in database-driven applications

One major takeaway was the ease with which a poorly written query can introduce critical vulnerabilities. Writing insecure code with raw string concatenation served as a useful contrast to the more robust, parameterized approach. I also came to appreciate how a small amount of validation logic (e.g., checking for blank inputs) can make a significant difference in both user experience and code stability.

**Challenges Faced**

* **Refactoring Logic**: Updating raw SQL to parameterized statements required me to carefully update each function without breaking the overall structure.
* **SQLite Behavior**: SQLite’s loose typing system caused a few bugs during validation testing, especially with numeric fields and null values.
* **Expanding Functionality**: Implementing the search feature challenged me to revisit SQL’s pattern matching capabilities using LIKE, and to manage edge cases such as partial or case-sensitive matches.

Through this enhancement, I not only demonstrated my proficiency with SQLite and SQL syntax but also developed confidence in managing security, structure, and modularity in real applications. This artifact now better represents my ability to apply academic learning to build practical tools—a core skill for any computer science professional.