**5-2 Milestone Four: Database Enhancement**

Alysha Pursley

Southern New Hampshire University

CS499: Computer Science Capstone

Professor Ramsey Kraya

July 27, 2025

**5-2 Milestone Four: Database Enhancement**

**Artifact Overview**

The artifact selected for this enhancement is the same Corner Grocer application originally developed in C++ for CS 210: Programming Languages. In its earliest form, the application read grocery sales data from a text file, stored the frequency of each item using a std::map, and allowed users to view results as a list or a histogram. In earlier enhancements, the program was rewritten in Python and expanded with sorting and search capabilities. For this third enhancement, the focus shifted to replacing the program’s in-memory data handling with a persistent SQLite database backend. The change ensures that data is stored and retrieved efficiently, even across program restarts, without losing any of the program’s original functionality.

***Justification for Inclusion***

This enhancement was selected because it demonstrates my ability to integrate a database into an existing application while maintaining backward compatibility with all previously implemented features. In practical terms, this update improves the program’s value for real-world use by allowing it to store, retrieve, and update inventory frequency data directly from a database rather than relying solely on memory. The use of SQLite means the application can be run as a standalone program without requiring additional database setup, making it portable and easy to distribute. Additionally, this enhancement showcases skills in database schema design, SQL query writing, and Python database API usage, all of which are important for full-stack and backend development roles.

***Skills and Course Outcomes***

1. **Course Outcome 1:**

“Employ strategies for building collaborative environments that enable diverse audiences to support organizational decision-making in the field of computer science.”

Some of the skills I’ve demonstrated in this enhancement that meet this outcome are:

1. Modular Designy
2. Code Documentation
3. Modular Architecture
4. Code Readability
5. Separation of Concerns

**How these skills support this outcome:**

* **Modular Design –** By placing all database-related logic into a dedicated db\_handler.py file, the project is easier for multiple developers to work on without interfering with unrelated code.
* **Code Documentation –** Structured, descriptive comments clearly explain the purpose of each SQL operation and why certain database practices were used, making it easier for collaborators to understand the database layer.
* **Modular Architecture –** Separating database operations from the main menu code allows each part to be developed or updated independently, supporting teamwork.
* **Code Readability –** Using clear function names like connect\_to\_db and insert\_item makes it obvious what each part of the program does.
* **Separation of Concerns –** Database logic is fully isolated from user interaction logic, reducing the chance that changes to one will break the other.

1. **Course Outcome 2:**

“Design, develop, and deliver professional-quality oral, written, and visual communications that are coherent, technically sound, and appropriately adapted to specific audiences and contexts.”

Some of the skills I’ve demonstrated in this enhancement that meet this outcome are:

* Menu Design
* User Experience Improvements
* Expanded Menu System
* Code Documentation
* Code Readability

**How these skills support this outcome:**

* **Menu Design –** The main menu clearly labels all options related to database actions, making them intuitive for end users.
* **User Experience Improvements –** The program now provides success messages when connecting to the database, inserting records, or retrieving data, as well as clear error messages if something goes wrong.
* **Expanded Menu System –** New menu options for loading data into the database and retrieving stored information are integrated without overwhelming the user.
* **Code Documentation –** Comments explain how each database interaction works, helping others understand the application’s data flow.
* **Code Readability –** The logical organization and naming conventions make the code accessible to both technical reviewers and less experienced developers.

1. **Course Outcome 3:**

“Design and evaluate computing solutions that solve a given problem using algorithmic principles and computer science practices and standards appropriate to its solution while managing the trade-offs involved in design choices.”

Some of the skills I’ve demonstrated in this enhancement that meet this outcome are:

* Database Query Logic
* Separation of Concerns
* Function Abstraction
* Code Reusability

**How these skills support this outcome:**

* **Database Query Logic –** I implemented efficient SQL queries to store and retrieve grocery item frequencies, ensuring minimal overhead and accurate results.
* **Separation of Concerns –** Database CRUD operations are handled in db\_handler.py while presentation logic stays in the main program, keeping responsibilities clear.
* **Function Abstraction –** Each database operation is encapsulated in a dedicated function, making them easy to maintain and test individually.
* **Code Reusability –** The same database query functions can be used across different menu options without rewriting logic.

1. **Course Outcome 4:**

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

Some of the skills I’ve demonstrated in this enhancement that meet this outcome are:

* Programming Proficiency
* Database Integration
* Function Abstraction
* Code Reusability
* Preserved Backward Compatibility

**How these skills support this outcome:**

* **Programming Proficiency –** The enhancement demonstrates strong Python and SQL skills by integrating SQLite and writing clean, functional database code.
* **Database Integration –** Introducing persistent storage into the application transforms it from a temporary data processor to a fully capable inventory tracking system.
* **Function Abstraction –** Isolating SQL operations into functions makes the database layer easier to expand in the future.
* **Code Reusability –** The database functions can be adapted for other projects requiring similar functionality.
* **Preserved Backward Compatibility –** All existing non-database features still work, ensuring that the enhancement adds value without removing existing functionality.

1. **Course Outcome 5:**

“Develop a security mindset that anticipates adversarial exploits in software architecture and designs to expose potential vulnerabilities, mitigate design flaws, and ensure privacy and enhanced security of data and resources.”

Some of the skills I’ve demonstrated in this enhancement that meet this outcome are:

* Input Validation
* Error Handling Enhancements
* User Experience Improvements

**How these skills support this outcome:**

* **Input Validation –** Only valid grocery item names are accepted, preventing injection of invalid or potentially harmful data into the database.
* **Error Handling Enhancements –** Database operations include try/except blocks to handle connection failures or query errors without crashing the application.
* **User Experience Improvements –** Error messages are clear and informative, helping users correct mistakes without compromising data integrity.

***Reflection on the Enhancement Process***

This enhancement significantly changed the internal workings of the Corner Grocer application while keeping the user interface consistent with earlier versions. Moving the data into a SQLite database required rethinking how the program reads, stores, and retrieves information. One of the main lessons learned during the process was the importance of separating business logic from database operations, which makes the program more modular and easier to maintain. I also learned the value of user feedback in database-driven applications—providing clear connection messages, success confirmations, and error outputs makes the system feel more reliable and professional to the end user.

The biggest challenge was ensuring backward compatibility while shifting the program from file-based data storage to a persistent database. This required carefully mapping the existing data flow to new database operations and making sure that all existing features still worked exactly as they did before. Error handling was another important focus; the program now gives clear error messages when the database cannot be connected to or when a file is missing, which would have been silent failures in earlier versions. Overall, this enhancement strengthened the application’s reliability, maintainability, and professional quality, making it a stronger example of my abilities in database integration and software design.