5-2 Milestone Four Enhancement Three Databases

# Database Artifact

URL: <https://github.com/oluwasegunsnhu/CS499_Artifact_3/>

The original code for the inventory management system contains several issues that can impact performance, security, and overall functionality. Let's examine these issues in detail:

### Lack of Connection Closure:

In the original code, the database connection is not closed explicitly. Failing to close the connection can lead to resource leaks, which can negatively impact performance and scalability. Leaving connections open for extended periods can exhaust the available connection pool and result in performance degradation and potential errors.

### String Concatenation for SQL Queries:

The original code uses string concatenation to incorporate input values directly into SQL queries. This approach is highly vulnerable to SQL injection attacks, where malicious users can exploit the code by injecting unauthorized SQL commands. SQL injection attacks can compromise data integrity, confidentiality, and even lead to the unauthorized execution of arbitrary SQL statements. It is crucial to use parameterized queries or prepared statements to mitigate these risks.

### Lack of Transaction Management:

The original code does not utilize transactions when modifying the database. Transactions ensure atomicity, consistency, isolation, and durability (ACID properties) for database operations. Without proper transaction management, errors or interruptions during database updates could leave the database in an inconsistent state. For example, if an error occurs while adding a product, the quantity update may fail, leading to discrepancies in the database.

### Absence of Error and Exception Handling:

The original code lacks proper exception handling. Exceptions provide a mechanism to gracefully handle errors, recover from exceptional situations, and provide informative feedback to users. Without exception handling, unexpected errors or database issues could crash the program or leave the user unaware of the problem, resulting in a poor user experience.

### Limited Search Capability:

The search functionality in the original code only supports exact matching of product names. It does not provide flexibility for partial matches or case-insensitive searches. This limitation can hinder the usability and efficiency of the inventory management system, especially when dealing with many products.

### Lack of Input Validation:

The original code does not perform sufficient validation of user inputs. It assumes that the inputs are valid and trusted. Failing to validate inputs can lead to various issues such as data corruption, integrity violations, or incorrect database operations. It is essential to validate and sanitize user inputs to ensure data integrity and protect against malicious or erroneous inputs.

These issues in the original code can have significant implications on performance, security, and functionality of the inventory management system. It can lead to resource leaks, expose the system to SQL injection attacks, cause data inconsistencies, provide a poor user experience, and leave the system vulnerable to errors and misuse.

# New Improvements:

The updated code for the inventory management system addresses the issues present in the original code and introduces several improvements.

### Connection Closure:

The new code properly handles the closure of the database connection by implementing the **\_\_del\_\_()** method in the InventoryDatabase class. This ensures that the connection is closed when the object is destroyed, preventing resource leaks, and promoting efficient resource utilization. Closing the connection helps maintain a healthy connection pool and avoids potential performance degradation.

### Parameterized Queries:

The new code utilizes parameterized queries throughout the codebase. It uses placeholders in the SQL statements and passes the input values as parameters to the **execute()** method. This approach mitigates the risk of SQL injection attacks by separating the SQL code from the user input. Parameterized queries ensure that user inputs are treated as data and not as executable code, making the system more secure and safeguarding against unauthorized database access.

### Transaction Management:

The new code does not explicitly handle transactions, but it allows exceptions to propagate up the call stack. This allows for better transaction management at a higher level of the code or application. By utilizing proper transaction management techniques, such as beginning a transaction, committing changes, or rolling back in case of errors, you can ensure data consistency and integrity. Transactional behavior helps prevent data discrepancies and ensures that database modifications are performed atomically.

### Basic Exception Handling:

Although the new code does not include explicit exception handling, it allows exceptions to be caught and handled by the calling code. This provides an opportunity to implement appropriate error handling strategies based on specific requirements. By allowing exceptions to propagate up the call stack, it enables better error reporting and handling, ensuring that users are informed about any issues that occur during database operations.

### Enhanced Search Capability:

The new code retains the search functionality but improves it by using a parameterized query with a LIKE operator. This enables partial matching of product names and supports case-insensitive searches. By incorporating wildcard characters in the search query, users can search for products more flexibly, enhancing the usability and efficiency of the system.

### Input Validation:

While not explicitly implemented in the updated code, input validation is an essential aspect that should be incorporated in real-world applications. Input validation ensures that user inputs meet specific criteria, such as data type validation, length checks, or format validation. Implementing input validation guards against erroneous or malicious inputs, protects data integrity, and provides a more robust and secure system.

The new code introduces significant improvements by closing the connection properly, implementing parameterized queries, addressing transaction management at a higher level, allowing for better exception handling, enhancing search capabilities, and emphasizing the importance of input validation. These improvements enhance the performance, security, functionality, and user experience of the inventory management system.

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

The artifact is a Python code for an inventory management system using SQLite database. It was originally created for “Database Management” course. It includes functions for adding products, displaying the product list, updating product quantity, and searching for products by name. The improved artifact is the updated version of older database which focuses to resolve several issues.

## 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 algorithms and data structure? How was the artifact improved?

I selected this artifact for inclusion in my ePortfolio because it demonstrates my skills in database management and interaction using Python. The original code had several issues related to connection management, SQL injection vulnerabilities, lack of transactions, and exception handling. I improved the artifact by addressing these issues, introducing connection closure, parameterized queries, transaction management at a higher level, and basic exception handling. These improvements showcase my ability to enhance code functionality, improve security, and ensure better performance and reliability.

## Did you meet the course objectives you planned to meet with this enhancement in Module One? Do you have any updates to your outcome-coverage plans?

With the enhancements made to the artifact in Module One, I have met several course objectives related to database management, security best practices, and code optimization. The improvements align with the objective of demonstrating proficiency in developing robust and secure software systems.

## Reflect on the process of enhancing and/or modifying the artifact. What did you learn as you were creating it and improving it? What challenges did you face?

During the process of enhancing and modifying the artifact, I learned the importance of proper connection closure, parameterized queries, and transaction management for maintaining data integrity and security. I also gained insights into the significance of exception handling to ensure better error reporting and user experience. The challenge I faced was identifying and addressing the specific issues in the original code, particularly related to SQL injection vulnerabilities and transaction management. However, through research and careful implementation, I was able to overcome these challenges and improve the functionality and security of the inventory management system.