# Android Application Enhancement

URL: <https://github.com/oluwasegunsnhu/CS499_Artifact_1/>

Previous code for designed for an independent mobile app with minimum User Interface. However, it did not include any database integration, editing functionalities, Error Handling, and several coding standards. In the new code these features were added to increase the performance and efficiency of the mobile application. The improvements made to the code were necessary to enhance the functionality, user experience, and performance of the inventory management app. The introduction of data persistence using an SQLite database ensures that the product list remains intact across app sessions, providing a seamless and reliable experience for users. The addition of editing functionality allows users to modify existing products, offering more flexibility and control over their inventory. Separating the add and edit functionality into separate dialog classes improves code organization and modularity, making it easier to maintain and extend the app in the future. Improved error handling prevents crashes and unexpected behavior when handling user input. Furthermore, the performance improvements, such as optimized data storage and retrieval, separation of concerns, and efficient UI rendering, enhance the app's overall performance, ensuring smooth and responsive user interactions even with a large number of products.

# New Improvements:

The improvements made to the code were crucial to address both logic and performance aspects of the inventory management app. From a logic perspective, the introduction of data persistence through a SQLite database ensures that the product list remains intact across app sessions, preventing data loss and providing a seamless user experience. The addition of editing functionality allows users to modify existing products, enhancing the app's flexibility and usability. Moreover, separating the add and edit functionalities into separate dialog classes improves code organization, making it easier to understand and maintain in the long run.

In terms of performance, the utilization of a SQLite database enhances data storage and retrieval efficiency, optimizing the app's overall performance. Storing the product list in a database ensures faster and more efficient data management compared to storing it solely in memory. The implementation of ListView.builder for rendering the product list ensures smooth scrolling performance, even with a substantial number of products. Furthermore, the separation of concerns and optimized database operations minimize unnecessary overhead, resulting in a faster and more responsive user interface. These improvements are briefly listed below:

1. **Logic Improvements:**

a. **Data Persistence:** The inclusion of a SQLite database allows for persistent storage of the product list. This ensures that the product data is retained even when the app is closed and reopened. It provides a more reliable and consistent user experience.

b. **Editing Functionality:** The ability to edit existing products enhances the app's functionality. Users can modify product details directly, improving the usability and flexibility of the inventory management system.

c. **Separation of Dialogs:** Separating the add and edit functionality into separate dialog classes improves code organization and modularity. Each dialog class handles a specific task, making the code easier to understand, maintain, and extend.

d. **Improved Error Handling:** The code now includes error handling for parsing the quantity input in the dialogs. This prevents potential crashes or unexpected behavior when invalid input is provided, ensuring a more robust and stable application.

1. **Performance Improvements:**

a. **Data Persistence Efficiency:** By integrating a SQLite database, the new code achieves efficient data persistence. Storing the product list in a database allows for optimized data retrieval and storage operations, resulting in better performance compared to storing the data in memory alone.

b. **Separation of Concerns:** The separation of database operations into separate methods improves performance by maintaining a clear distinction between data management and user interface-related tasks. It allows for efficient database operations without unnecessary overhead in UI rendering.

c. **Optimized ListView Rendering:** The use of **ListView.builder** efficiently renders only the visible items in the product list, ensuring smooth scrolling performance even with a large number of products. The **ListView.builder** widget lazily creates and disposes of list items as needed, reducing memory consumption and improving overall performance.

d. **Database Query Optimization:** The use of SQLite database queries for saving and loading the product list optimizes data retrieval and storage. The new code retrieves the product list from the database only when required, rather than loading all data on every app start. This optimization improves performance, especially when dealing with a large number of products.

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

The original artifact was created for a course project for “Mobile Development” course. This inventory management app had limited features and it could not be used for real-world applications. This code snippet representing an inventory management system. It was created as an enhanced version of an existing basic code, incorporating improvements in security, modularity, performance, error handling, and database connection management. The artifact was developed in the context of an ePortfolio to showcase software development skills and abilities.

## 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?

The inclusion of this artifact in the ePortfolio is justified because it demonstrates the ability to enhance and optimize existing code, apply best practices in software development, and showcase proficiency in database management, object-oriented design, error handling, and code structure. The specific components of the artifact that highlight these skills include the encapsulation of database operations in a class, the implementation of parameterized queries for security, the use of try-except blocks for error handling, and the proper management of database connections.

The artifact was selected to showcase the ability to take an existing codebase and improve upon it, addressing issues and enhancing performance. The enhancements made in the artifact, such as implementing parameterized queries and connection management, demonstrate a proactive approach to software development and a commitment to quality and best practices.

## 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?

Yes, the enhancements made in Module One align with the course objectives planned to be met. The improvements in security, modularity, performance, and error handling directly address the objectives of enhancing code quality, applying best practices, and demonstrating proficiency in software development principles. There are no updates to the outcome-coverage plans as the enhancements made in the artifact align with the initial objectives.

## 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?

It was a valuable learning experience. I gained a deeper understanding of database management, object-oriented design, and the importance of code structure and modularity. I learned about the significance of parameterized queries for security and the challenges associated with database connection management. The process of addressing the issues in the basic code and implementing the improvements required careful analysis, research, and troubleshooting. Challenges included finding the most efficient and effective approaches to enhance the code while considering performance, security, and maintainability. However, these challenges provided an opportunity to develop problem-solving skills and gain a better understanding of software development best practices.