# Optimization, Scaling, and Final Evaluation Report

Name

Institution Affiliation

# Optimization, Scaling, and Final Evaluation Report

**1. Optimization Techniques**

A number of changes have been made on the optimized inventory management system in order to improve the model’s performance and scalability (Villegas-Ch et al., 2024). The following are some of the changes that can lead to better performance. First in order to improve the performance of the model, there is efficient data structure which has been achieved through the hash table which is used for quick SKU-based lookups which reduces the time it takes to retrieve data to O(1).

The second improvement is using the sorted dictionary (BST-like structure). This change is used to ensure there is fast price-based search. The Min-Heap also allows for an efficient data retrieval of low-stock items with O(log n) complexity. Lastly, the Deque is used to optimize queue operations for restocking using O(1) insertions and deletions.

Results

Inventory after adding products:

[Product(SKU: A100, Price: 20.5, Category: Electronics, Stock: 15), Product(SKU: B200, Price: 10.0, Category: Groceries, Stock: 5), Product(SKU: C300, Price: 50.0, Category: Clothing, Stock: 2), Product(SKU: D400, Price: 15.0, Category: Books, Stock: 20)]

Fetching product B200:

Product(SKU: B200, Price: 10.0, Category: Groceries, Stock: 5)

Inventory after removing product A100:

[Product(SKU: B200, Price: 10.0, Category: Groceries, Stock: 5), Product(SKU: C300, Price: 50.0, Category: Clothing, Stock: 2), Product(SKU: D400, Price: 15.0, Category: Books, Stock: 20)]

Processing Restock:

Product(SKU: B200, Price: 10.0, Category: Groceries, Stock: 5)

Product with lowest stock:

Product(SKU: C300, Price: 50.0, Category: Clothing, Stock: 2)

Products in price range $10 - $25:

[Product(SKU: B200, Price: 10.0, Category: Groceries, Stock: 5), Product(SKU: D400, Price: 15.0, Category: Books, Stock: 20)]

In [ ]:

Reduction of Redundant Operations

Here the improvements are made by removing the elements from the heap when inventory is updated. It also includes managing the memory efficiently. This is done by ensuring that elements are removed from the structure when deleted.

2. Scaling Strategy

Scaling strategy ensures that the model can handle larger datasets. This system will be able to manage a lot of inventory items using O(log n) or better operations for most queries. It also uses a sorted dictionary and thus avoiding costly dataset re-sorting which can be costly.

Memory Optimization

There is memory optimization because of the use of efficient in-memory data structure which minimizes duplication. There is also dynamic memory allocation and therefore only the most relevant data is stored in the memory. The model considers parallel processing ensuring that the system supports multithreading which can run concurrent stock updates.

3. Testing and Validation

Test cases include

The test case begins by adding and removing products to validate SKU uniqueness and retrieval. We also query products based on price ranges which handling restocking operations more effectively. Lastly we extract the lowest-stock product to verify heap efficiency.

Stress Testing

For stress testing, we begin by loading the system with over 100,000 products to verify performance under scale ensuring that retrieval times remain O(1) for lookups and O(log n) for priority-based queries.

**4. Performance Analysis**

|  |  |  |
| --- | --- | --- |
| Operation | Initial Complexity | Optimized Complexity |
| SKU Lookup | O(n) | O(1) |
| Price-based Query | O(n log n) | O(log n) |
| Low-stock Retrieval | O(n) | O(log n) |
| Restock Processing | O(n) | O(1) |

The optimized model we developed above will be able to improve the overall efficiency by reducing overhead, ensures quick product access, and scales effectively as data volume grows.

5. Final Evaluation

The final model we created is an optimized inventory system that improves performance because lookups are now faster and low memory consumption. We have also improved the scalability of the model which making it easier to handle large datasets more efficiently. The robust testing and validation ensures the correctness of the system under stress conditions.

Future Improvements

Future improvements may include things like adding database-backed persistence for data storage (Yang et al., 2018). We can also add multithreaded stock update to improve their efficiency and time. Enhancing the user interface by having a real-time dashboard to provide statistical insights for customers and relevant data for decision making might also help you enhance the model.

# References

Yang, J., Subramaniam, P., Lu, S., Yan, C., & Cheung, A. (2018, May). How not to structure your database-backed web applications: a study of performance bugs in the wild. In *Proceedings of the 40th International Conference on Software Engineering* (pp. 800-810).

Villegas-Ch, W., Navarro, A. M., & Sanchez-Viteri, S. (2024). Optimization of inventory management through computer vision and machine learning technologies. *Intelligent Systems with Applications*, *24*, 200438.