Analysis

1. Analyze the time complexity of each operation (add, update, delete) in your chosen data structure.

Ans. Here in my application, I have used Hashmap as my data structure to implement the project. This has been further explained below:

* Add Operation

Operation: Adding a product to the Inventory HashMap.

Time complexity: O(1)

Hash maps provide constant-time performance for insertions under typical conditions.

* Update Operation

Operation: Updating a product in the Inventory HashMap.

Average Case: O(1)

Updating an existing entry involves a hash map insertion, which is O(1) on average.

* Delete Operation

Operation: Removing a product from the HashMap.

Average Case: O(1)

Removing an entry from a hash map typically takes constant time.

As the id has been generated dynamically here, there is no occurrence of hash collisions so all the operations works in constant time complexity of O(1).

1. Discuss how you can optimize these operations.

Ans.

* Avoiding Unnecessary Operations

Optimization: Check if the product exists before updating to avoid redundant put operations.

* Batch Processing

Optimization: If possible, perform operations in batches to reduce the overhead associated with individual operations. This can be particularly effective when loading a large number of products initially.

* Allowing system to take id as Input

Optimization: It is more user friendly to take the id as input, but that would generate the possibility of hash collisions. Proper function would then be needed to solve them to reduce time complexity.

By implementing these optimizations, you can ensure that the inventory management system performs efficiently even as the size of the inventory grows.