**Task -1 : Inventory Management System**

Solution by: Mahika Thakur, CSE 7TH Semester, MCKV Institute of Engineering

**Q. Explain why data structures and algorithms are essential in handling large inventories.**

Data structures and algorithms are crucial in managing large inventories because they allow efficient storage, retrieval, and manipulation of data. Without appropriate data structures, operations such as adding, updating, and deleting products can become time-consuming and resource-intensive, leading to inefficiencies and potential system slowdowns. Effective use of algorithms ensures that operations are performed optimally, reducing the time complexity and improving the overall performance of the inventory management system.

Types of Data Structures Suitable for This Problem:

1. ArrayList: Useful for sequential access but not ideal for frequent insertions and deletions.
2. HashMap: Ideal for storing products with unique identifiers (e.g., productId). Provides average O(1) time complexity for insertion, deletion, and lookup operations, making it highly efficient for large inventories.
3. LinkedList: Useful for frequent insertions and deletions, but access time is slower compared to ArrayList and HashMap.

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

Time Complexity Analysis

Add Product:

Best Case: O(1) - When the product ID is unique and can be added directly.

Average Case: O(1) - HashMap operations typically have O(1) average time complexity.

Update Product:

Best Case: O(1) - When the product ID exists, and the update is straightforward.

Average Case: O(1) - Updating a value in a HashMap is an O(1) operation.

Delete Product:

Best Case: O(1) - When the product ID exists and can be removed directly.

Average Case: O(1) - Deleting a value from a HashMap is an O(1) operation.

**ii)Discuss how you can optimize these operations**

The chosen HashMap (dictionary) data structure already provides optimal average-case time complexity for add, update, and delete operations. However, the following optimizations can be considered:

Lazy Deletion: Instead of deleting a product immediately, mark it as inactive. This can be useful in scenarios where deletion operations are frequent and later undo operations are required.

Batch Operations: Implement methods to handle batch updates or deletions to reduce the overhead of multiple individual operations.