Technical Report: Data Structures in Inventory Management System

Project: Saaman (PIRS)

1. Executive Summary

This project ("Saaman") utilizes a specific set of foundational Data Structures and Algorithms (DSA) to solve distinct logistical problems. Unlike a standard CRUD application that relies solely on database queries, this system implements in-memory structures to optimize for **speed**, **priority handling**, and **simulation logic**.

2. Data Structure Analysis

2.1. Hash Map (Dictionary)

File: data_ingestion.py
Implementation: Python Dictionary (dict)

Usage: The get_product_lookup() function creates a mapping of SKU -> Product Details. Used throughout the app to instantly retrieve a product's Name, Price, and Stock Level given its SKU.

Why it is needed: In a warehouse, scanning a barcode (SKU) must return product details instantly. A linear search through a list would be too slow (O(n)). A Hash Map provides constant time access.

Complexity: Time: O(1) (Average), Space: O(n)

2.2. Min-Heap (Priority Queue for Reorders)

File: prioritization.py
Implementation: Python heapq module

Usage: Stores products as tuples: (Stability Score, SKU). The 'Score' represents days remaining. The heap ensures the product with the **lowest** score (running out soonest) is always at the root.

Why it is needed: Managers need to know the *most critical* item to reorder immediately. A sorted list would require O(n log n) to rebuild every time stock changes. A Min-Heap allows us to peek at the most urgent item in O(1) and update/insert in O(log n).

Complexity: Time: O(log n) (Insert/Pop), O(1) (Peek Min), Space: O(n)

2.3. Max-Heap (Priority Queue for Shipping)

File: floor_operations.py (Class: ShippingQueue)
Implementation: Python heapq (stored with negative values)

Usage: Manages the outbound shipping lane. Prioritizes orders based on: **Expiration Date (FEFO)** > **Customer Tier (VIP)** > **Order Value**.

Why it is needed: A standard FIFO queue (First-In, First-Out) is inefficient if a VIP customer orders or if goods are about to expire. The Max-Heap ensures that high-priority orders "jump the line" dynamically.

Complexity: Time: O(log n) (Insert/Pop), Space: O(n)

2.4. Binary Search Tree (BST)

File: reporting.py (Class: InventoryBST)
Implementation: Custom Node and InventoryBST classes.

Usage: Organizes inventory explicitly by "Days Remaining". Used to generate the "Stability Report", separating items into "Critical" (Left Subtree) and "Stable" (Right Subtree).

Why it is needed: It maintains a structured order of inventory stability. It allows for efficient range queries (e.g., "Find all items with < 15 days stock") without iterating an unsorted array.

Complexity: Time: O(log n) (Average), Space: O(n)

2.5. Circular Linked List

File: reporting.py (Class: AuditList)
Implementation: Custom AuditNode and pointer logic.

Usage: Manages the daily audit schedule for warehouse shelves. The last node points back to the head.

Why it is needed: Audits are a never-ending process. Once a worker finishes checking the last shelf, they must immediately start again at the first. A circular list models this infinite loop perfectly without needing to reset an index.

Complexity: Time: O(1) (Move to Next), Space: O(n)

2.6. Hash Set (Set)

File: floor_operations.py (Class: SafetyCheck)
Implementation: Python set()

Usage: Stores "Blocked" or "Recalled" lot numbers. Checks membership before shipping.

Why it is needed: Similar to the Hash Map, checking if a specific item is dangerous/recalled must be instant. A Set provides O(1) membership testing.

Complexity: Time: O(1), Space: O(n)

3. Scenario Walkthrough: "The Journey of an Order"

Scenario: A VIP customer places an order for "Wireless Mouse". We have 50 in stock.

Step	Action	Data Structure	Logic
1	Order Validation	Hash Map	Instant lookup of price & existence (O(1)).
2	Prioritization	Max-Heap	Order scored +30 for VIP. Inserted into heap (O(log n)). Jumps ahead of s
3	Inv. Update	Min-Heap	Stock drops to 49. New priority calculated. Min-Heap re-balanced to reflec
4	Audit	Circular Linked List	Manager moves audit pointer to next shelf. (O(1)).
5	Safety Gate	Hash Set	Worker picks Lot #99. System checks 'if Lot in BlockedSet' (O(1)). Safe.

4. Complexity Cheat Sheet

Data Structure	Role	Access/Search	Insertion	Deletion
Hash Map	Product Lookup	O(1)	O(1)	O(1)
Min-Heap	Reorder Priority	O(1) (Min)	O(log n)	O(log n)
Max-Heap	Shipping Queue	O(1) (Max)	O(log n)	O(log n)
BST	Stability Sorting	O(log n)	O(log n)	O(log n)
Circular List	Audits	O(n)	O(1)	O(1)
Hash Set	Safety Checks	O(1)	O(1)	O(1)