SMART INVENTORY TRACKER

PROJECT REPORT

Contents

[1.Problem statement overview 3](#_Toc202275084)

[2.thought process and approach 4](#_Toc202275085)

[3.data structures used 5](#_Toc202275086)

[4.time and space complexity analysis 6](#_Toc202275087)

[5.how to run application 7](#_Toc202275088)

[6.sample input output commands 8](#_Toc202275089)

[7.flow chart 9](#_Toc202275090)

# 1.Problem statement overview

The goal of this project is to design and implement a **Smart Inventory Tracker** tailored for a small retail store. The store requires a simple but efficient system to manage its product inventory digitally. The application should allow store staff to:

* **Add new products** to the inventory, along with their quantity and timestamp.
* **Remove products** that are no longer in stock or need to be updated.
* **Check the availability** of specific products instantly.
* **View the entire inventory** in a sorted and user-friendly format.

To ensure a smooth and responsive user experience, the system must be designed with **optimized data structures** that allow operations like adding, removing, and checking products to occur in **constant (O(1)) or near-constant time**. Additionally, it should support **sorted display** using efficient sorting mechanisms such as heaps.

The application aims to minimize manual errors, speed up inventory tasks, and provide clear, organized insights into stock levels—making it especially useful for small businesses with limited resources.

# 2.thought process and approach

In today’s world, businesses require systems that can efficiently manage large inventories with speed and accuracy. Our goal was to design a **Smart Inventory Tracker** that is fast, scalable, and user-friendly.

We focused on:

* **Speed**: Using efficient data structures for quick add, remove, and search operations.
* **Simplicity**: A clean interface so even non-technical users can use it easily.
* **Scalability**: Capable of handling large inventories without slowing down.
* **Reliability**: Handles invalid inputs and prevents crashes.

Python was chosen for its simplicity and flexibility.The result is a smart, responsive, and easy-to-use inventory system fit for modern business needs.

# 3.data structures used

The Smart Inventory Tracker project utilizes several core data structures from Python to manage and operate inventory data efficiently. These structures play a key role in supporting the system’s performance and logic.

**Array / List**

In the simpler implementation of the tracker, a Python list is used to store inventory items. Each item is typically stored as a tuple containing the product name and quantity, e.g., ("mango boxes", 5). Lists allow sequential storage and iteration but have linear time complexity when it comes to searching and removing specific elements.

**Dictionary**

In the enhanced version of the project, a Python dictionary is used as the primary structure to store inventory. Each product name acts as a key, with its value being another dictionary that holds both the quantity and the datetime the item was added. This structure allows fast access, updates, and deletions, with an average time complexity of O(1), making it ideal for efficient inventory tracking.

**Heap(heapq)**

To display inventory in sorted (alphabetical) order, the system uses a min-heap through Python’s built-in heapq module. Items are pushed into the heap and then popped out to generate a sorted list. This provides efficient sorting with a time complexity of O(n log n), leveraging the benefits of the heap data structure.

**String Handling (pyfiglet)**

To enhance the user interface, the pyfiglet library is used to render the application title in an ASCII art style. While this has no effect on the core logic, it improves user experience through better visual presentation, using Python's string handling and formatting capabilities.

**Datetime**

The datetime module is used to timestamp each item when it is added to the inventory. This allows the system to track when actions occur and provides visibility into when inventory status was last viewed or updated. These datetime objects are stored alongside inventory data for better traceability and reporting.

# 4.time and space complexity analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **OPERATION** | **FUNCTION** | **TIME COMPLEXITY** | **SPACE COMPLEXITY** |
| ADD ITEM | ADD\_ITEM() | O(1) | O(1) |
| REMOVE ITEM | REMOVE\_ITEM() | O(1) (DICT), O(N) (LIST) | O(1) |
| CHECK AVAILABILITY | IS\_AVAILABLE\_IN\_STOCK() | O(1) (DICT), O(N) (LIST) | O(1) |
| SORT INVENTORY | SORTED\_INVENTORY() (HEAPQ) | O(N LOG N) | O(N) |
| CLEAR INVENTORY | CLEAR\_INVENTORY() | O(1) | O(1) |
| SHOW STATUS | INVENTORY\_STATUS() | O(N) | O(1) |

**O(1) – Constant Time**

The operation takes the same amount of time no matter how much data there is.

**O(n) – Linear Time**

The time grows directly with the number of elements.

**O(n log n) – Log-Linear Time**

Slightly more than linear, used in efficient algorithms like heap sort or merge sort.

# 5.how to run application.

### **Requirements:**

* Python 3.x installed on your system.
* Required libraries:
  + ***pyfiglet***(for stylized title)
  + ***heapq*** (built-in, no install needed)
  + ***datetime*** (built-in, no installation needed)

You can install *pyfiglet* by running:

*pip install pyfiglet*

### **Steps to Run (Command-Line Version):**

1. Open **Command Prompt** (CMD).
2. Navigate to the project directory in your computer :

For example: *cd D: \DSA\projects\project1*

1. Run the Python script:

*python smart\_inventory\_tracker.py*

1. The program will launch in terminal with the following options:

SMART INVENTORY TRACKING SYSTEM

----------------------------------

1. Add Item

2. Remove Item

3. Check Availability

4. Sort Inventory

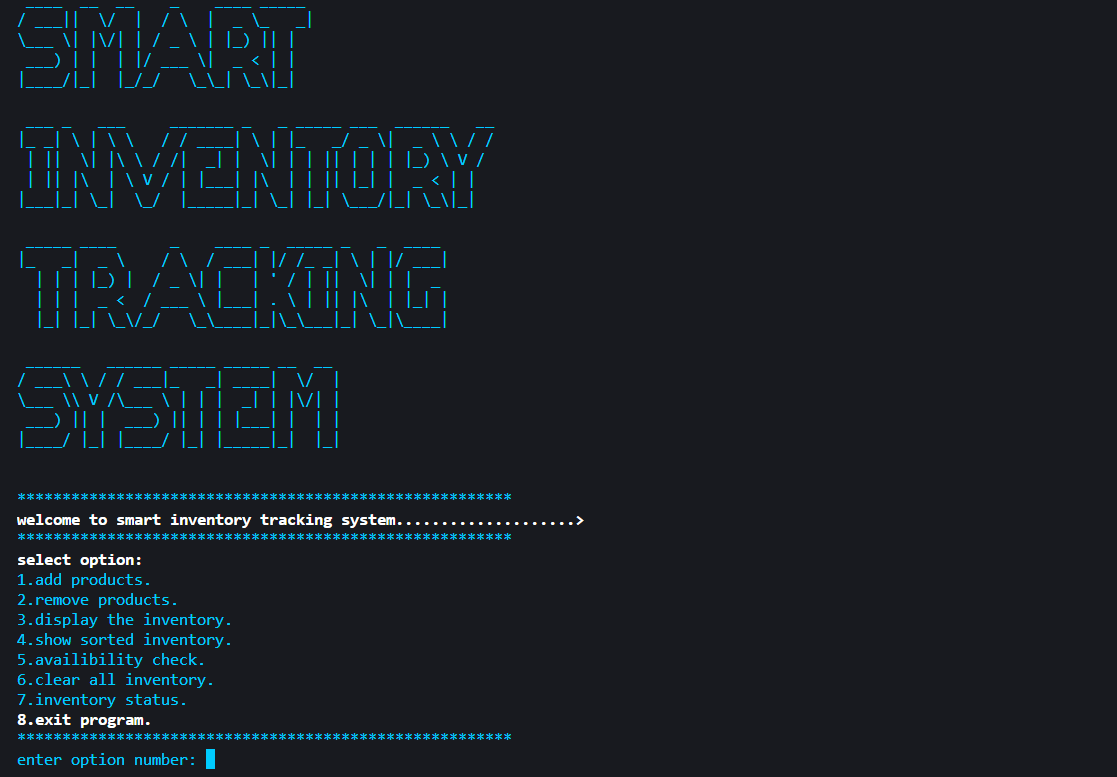
5. Clear Inventory

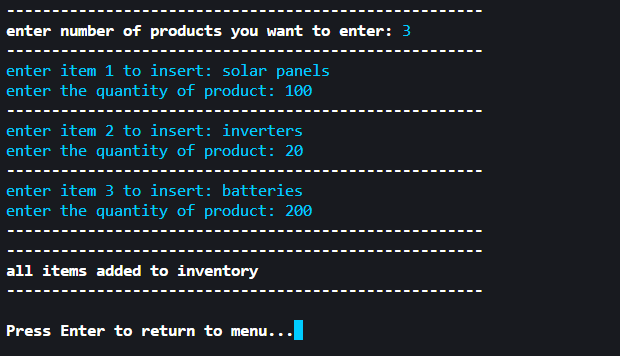
6. Show Status

7. Exit

1. Follow on-screen instructions to use the inventory system.

# 6.sample input output commands





A screen shot of a computer

AI-generated content may be incorrect.

# 7.flow chart

**START PROGRAM**

**DISPLAY TITLE**

**SMART INVENTORY TRACK**

**DISPLAY MENU**

**1–ADD 2–REMOVE**

**3–CHECK 4–SORT**

**5- AVAILIBILITY CHECK**

**6- STATUS**

**7–CLEAR**

**8–EXIT**

**ADD: STORE ITEM IN ARRAY**

**REMOVE: DELETE FROM ARRAY**

**CHECK: SEARCH IN ARRAY**

**SORT: USE HEAPQ TO SORT**

**STATUS: COUNT & TIMESTAMP**

**CLEAR: EMPTY ARRAY**

**DISPLAY: DISPLAY INVENTORY**

**LOOP OR EXIT(8)**

**PERFORM ACTION BASED ON**

**INPUT (USING OOP CLASS)**