**SOFTWARE ENGINEERING DEPARTMENT**

**DS&A SEMESTER PROJECT**

**Last date of Submission: 28 MAY 2025**

# Submitted to: MR SHAKEEL AHMED

**GROUP MEMBERS**

|  |  |
| --- | --- |
| **NAME** | **REGISTRATION NO** |
| **MOHAMMAD SAAD SHAFIQUE** | **4779-FOC/BSSE/F23** |
| **MUBASHIR NAZIR** | **4792-FOC/BSSE/F23** |
| **MUHAMMAD TAHIR** | **4795-FOC/BSSE/F23** |
| **MUZAMMIL KHALIQ** | **4824-FOC/BSSE/F23** |

**PROJECT REPORT**

**Project Introduction:**

* + ***PROJECT TITLE:*** INVENTORY & STOCK CONTROL SYSTEM
  + ***PROGRAMMING LANGUAGE:*** C++
  + ***PROGRAMMING TOOLS USED:*** VSCODE and GITHUB
  + ***APPLICATION DOMAIN:*** Businesses involving any kind of inventory & stock
  + ***DS&A CONCEPT USED:*** Linked Lists, Binary Search Trees
  + ***PROJECT FEATURES:*** Product Addition, Deletion, Display, Sorting and Supplier Addition, Deletion, Display and Stock Addition, Display, Sorting and also save and load all data.

**Project Objective:**

Our project, Inventory & Stock Control System, is designed to streamline the management of inventory and stock for businesses of any scale. It offers essential features such as product addition, deletion, updating, and display to ensure efficient product handling. In addition to inventory tracking, the system also incorporates stock control functionalities to monitor and manage product quantities effectively.

From a technical perspective, the system leverages key Data Structures & Algorithms (DS&A) concepts, including Linked Lists and Binary Search Trees, along with search algorithms like Linear Search and Binary Search, to ensure optimized data storage and retrieval.

**Data Structures and Algorithms Used:**

**1. Binary Search Tree (BST) – for Products**

We used a Binary Search Tree to store and manage product data. This allows efficient insertion, searching, and deletion of products based on their unique product ID. In-order traversal is used to display products in sorted order.

**2. Singly Linked List – for Suppliers and Stock**

We implemented singly linked lists to store supplier and stock records. This structure allows flexible insertion and traversal without the overhead of resizing, making it ideal for dynamic data like suppliers and stock entries.

**3. Bubble Sort – for Products and Suppliers**

Bubble Sort was applied to sort product and supplier arrays by their respective IDs in ascending order. It is simple to implement and useful for small datasets.

**4. Merge Sort – for Stock (by Quantity)**

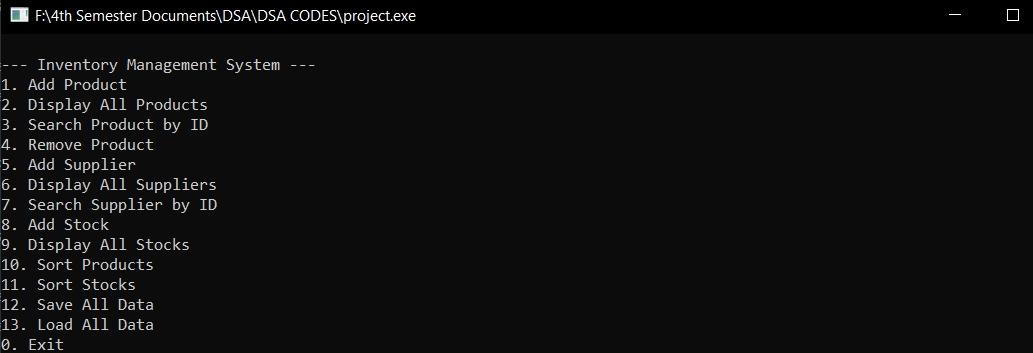
Merge Sort was used to sort stock records based on quantity in descending order. This algorithm is efficient for large datasets due to its divide-and-conquer approach and stable sorting.

**5. Linear and Binary Search**

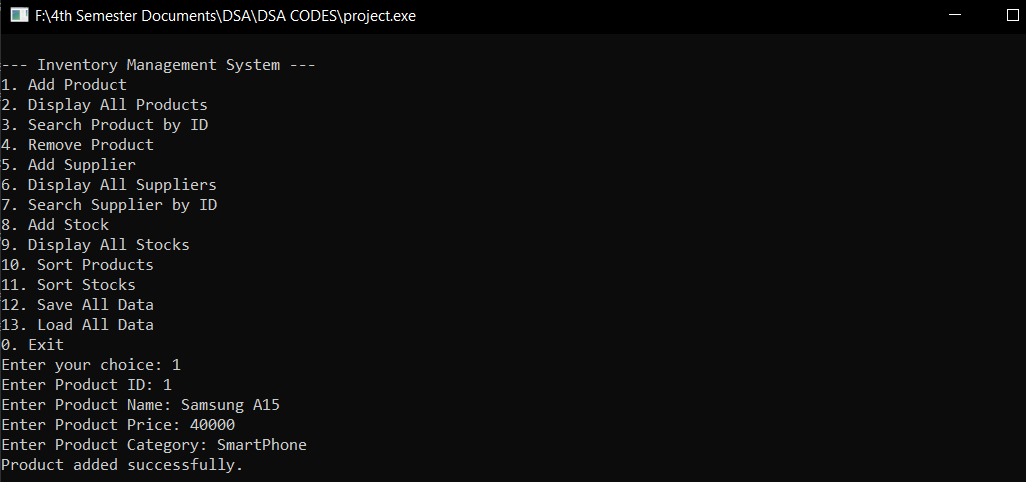
* **Linear Search** was used to find products and suppliers in unsorted arrays.
* **Binary Search** was applied on sorted arrays to quickly find entries using their IDs.

**Code Screenshots:**

**Project Main Interface:**

****

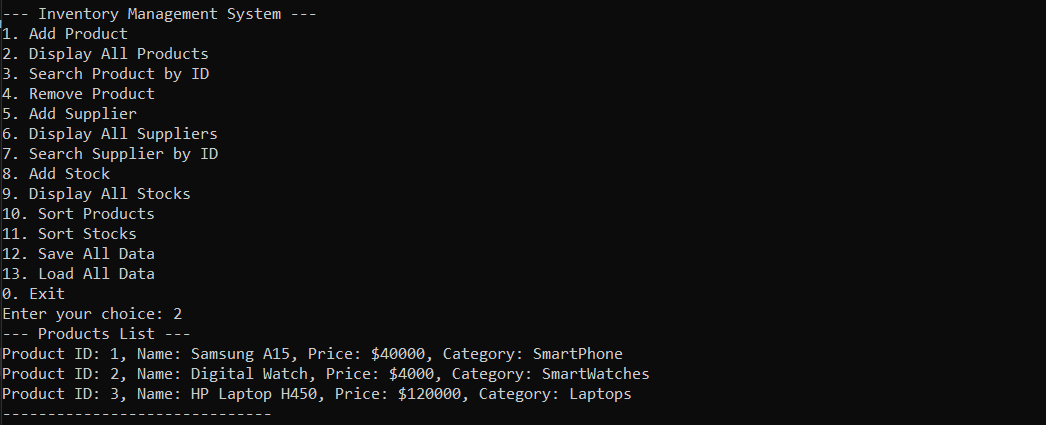
**Add Product:**

****

****

****

**Display Product:**

****

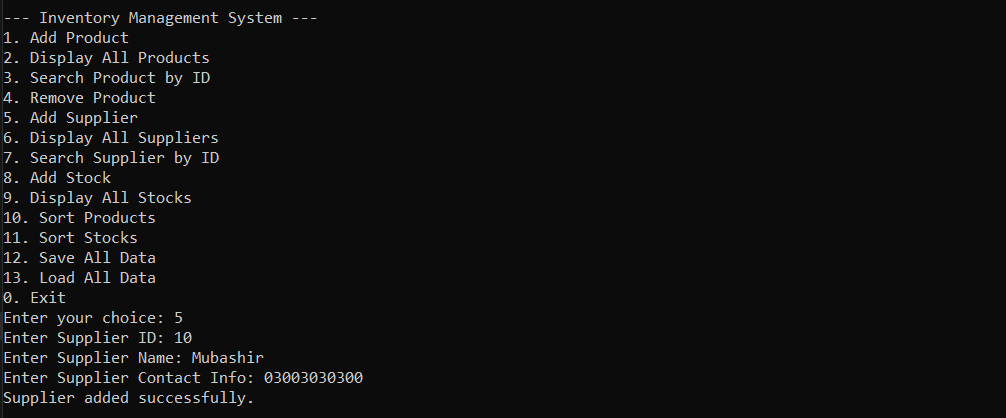
**Search Product by ID:**

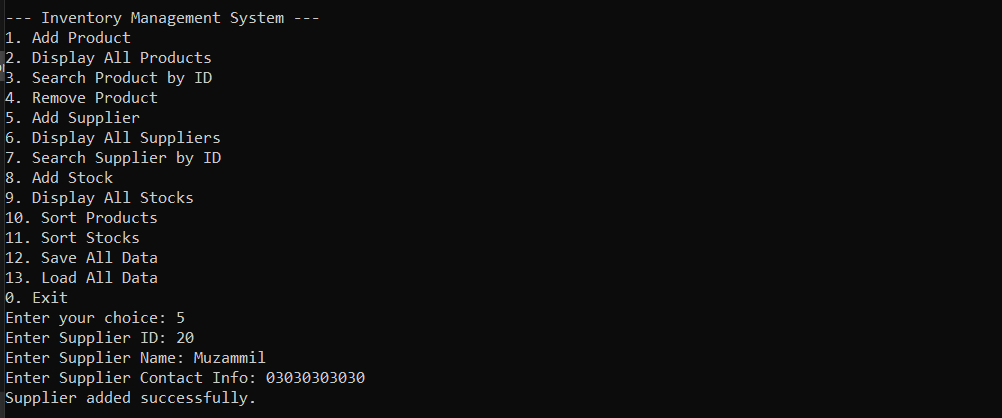
****

**Remove Product:**

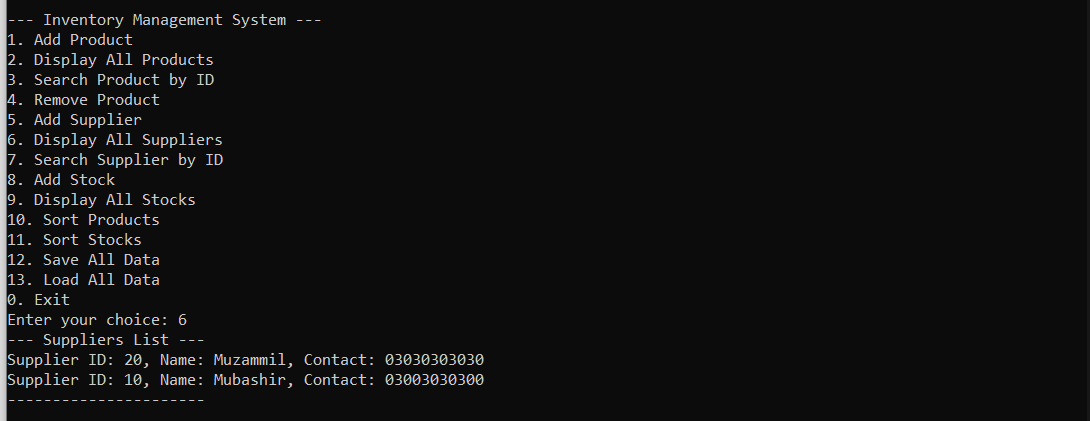
****

**Add Supplier:**

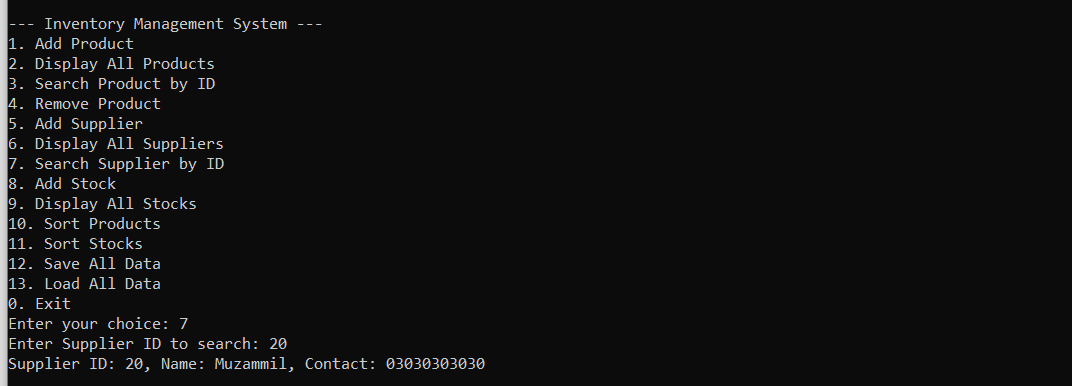
****

****

**Display All Supplier:**

****

**Search Supplier by ID:**

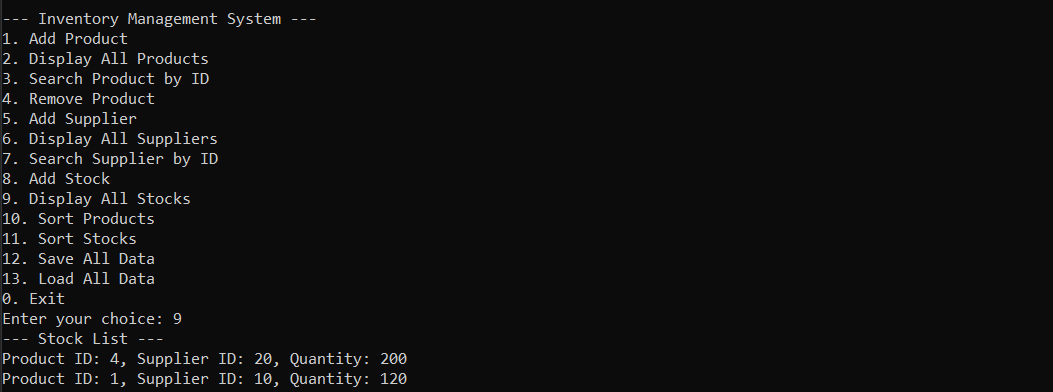
****

**Add Stock:**

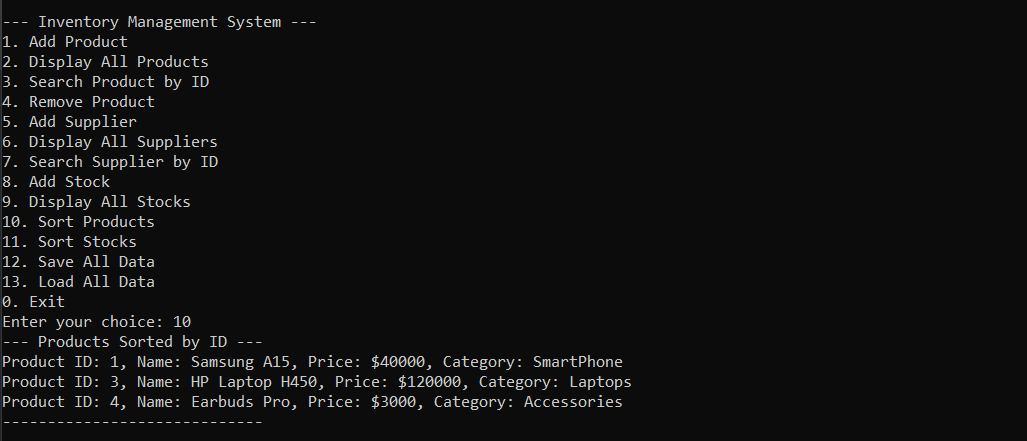
****

****

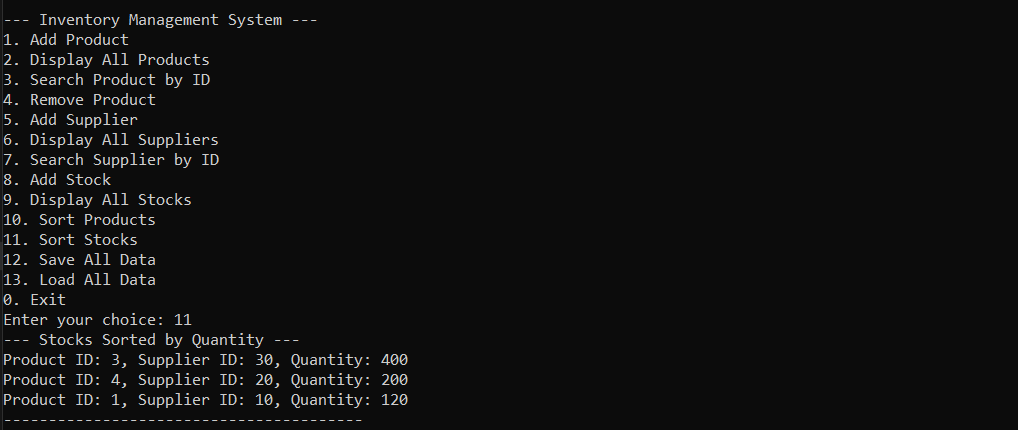
**Display All Stocks:**

****

**Sort Products by ID:**

****

**Sort Stocks by Quantity:**

****

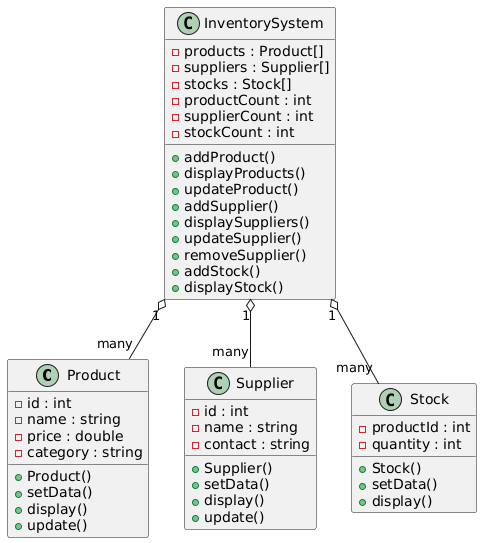
**Save all Data:**

****

**Load all Data:**

****

**UML Class Diagram:**

****

**Performance Analysis and Optimization Summary:**

**1. Binary Search Tree (BST) – Products**

* **Time Complexity**:
  + Insertion/Search/Deletion: Average **O (log n)**, Worst-case **O(n)** (if unbalanced)
* **Optimization**:
  + In-order traversal is used for displaying sorted products without needing external sorting.
  + Unique product ID ensures data consistency and efficient lookup.

**2. Singly Linked List – Suppliers and Stock**

* **Time Complexity**:
  + Insertion at head: **O (1)**
  + Searching: **O(n)**
* **Optimization**:
  + Linked list is used where order is less critical, and fast insertions are required without resizing.

**3. Sorting Algorithms**

| **Algorithm** | **Applied On** | **Time Complexity** | **Remarks** |
| --- | --- | --- | --- |
| **Bubble Sort** | Products, Suppliers | O(n²) | Simple to implement, used on small datasets |
| **Merge Sort** | Stock (by quantity) | O(n log n) | Efficient and stable for large data |

**4. Searching Algorithms**

| **Method** | **Use Case** | **Time Complexity** | **Remarks** |
| --- | --- | --- | --- |
| **Linear Search** | Unsorted arrays | O(n) | Simple, used where sorting is not guaranteed |
| **Binary Search** | Sorted arrays (by ID) | O(log n) | Fast lookup on sorted data |

**5. File Handling Optimization**

* Data is persisted using text files (products.txt, suppliers.txt, stocks.txt).
* On loading, duplicate ID exceptions are handled gracefully.
* Only necessary data is loaded and written, avoiding redundancy.

**Member Wise Contribution:**

1. **Mohammad Saad Shafique🡪 Coding + Professional Presentation**
2. **Mubashir Nazir🡪 Coding + Professional Presentation**
3. **Muhammad Tahir🡪 Coding + Design Documentation**
4. **Muzammil Khaliq🡪 Coding + Design Documentation**

**GitHub Repository Link:** [**GitHub Repository**](https://github.com/Mubi3784/Inventory-and-Stock-Control-System)

**LinkedIn Post Link:** [**LinkedIn Post**](https://www.linkedin.com/posts/mohammad-saad-shafique-532247332_dsa-semester-project-inventory-stock-activity-7333234464124211202-_1Je?utm_source=share&utm_medium=member_desktop&rcm=ACoAAFOvbjYB_qC2He-H_1lzHO3su4TjGUo35oc)

**Summary:**

The "Inventory and Stock Control System" is a C++-based console application developed as part of our Data Structures and Algorithms course. It allows users to manage products, suppliers, and stock quantities through efficient data structures and algorithms. Key features include product insertion/search/deletion using a Binary Search Tree (BST), supplier and stock management using singly linked lists, and data persistence through file handling.

The system demonstrates practical implementation of object-oriented principles, custom exception handling, and algorithmic logic, while offering a user-friendly, menu-driven interface.

**Conclusions:**

* The project successfully showcases the application of core data structures such as BSTs and linked lists in solving real-world problems.
* Sorting algorithms like Bubble Sort and Merge Sort were integrated effectively for data organization and performance.
* Searching was optimized using both linear and binary search techniques.
* File handling ensured data persistence between sessions, and exception handling added robustness to the system.
* The project met all academic requirements, including design, functionality, and documentation.

**Future Improvements:**

To enhance the system further, we propose the following upgrades:

* **User Authentication**: Add login/logout functionality for multi-user access and security.
* **Graphical User Interface (GUI)**: Convert the console-based application to a GUI using libraries like Qt or wxWidgets.
* **Database Integration**: Replace text files with a relational database (e.g., MySQL or SQLite) for better data management and scalability.
* **Self-Balancing Trees**: Use AVL or Red-Black Trees instead of BST to maintain balance and ensure optimal performance.
* **Reporting Module**: Generate reports on stock levels, supplier performance, and product movement.
* **Cloud Sync/Backup**: Store data in the cloud for access across systems and for automatic backup.
* **Mobile App**: Build a lightweight mobile version for real-time inventory access.