ಜಿ. ಎಂ. ವಿಶ್ವವಿದ್ಯಾಲಯ

GM UNIVERSITY

P. B. Road, Davanagere – 577 006 KARNATAKA | INDIA

Faculty of Engineering and Technology

School of Computer Science and Technology

Department of CSE- Data Science

Assignment Report – I

Subject Coordinator	Mrs.Ashwini G T	Semester/Section	3rd Sem/DS2A
Course	Object Oriented Programming	Course Code	UE24CS2303
CO Mapped	C05	Max Marks	10
Issue Date	9/11/25	Submission Date	24/11/25

Submitted By

Name of the Student : Nikitha SB

USN : U24E01DS049

COs	CO5	
Allotted Marks for COs	10	
Marks obtained for COs		
Submitted Date	Max Marks	Marks obtained
	10	

Academic Year: 2025-26 (ODD Semester)

Signature of the Student

Signature of the Subject Coordinator

CONTENTS :~

TOPIC :- INVENTORY AND STACK CONTROL SYSTEM

1. ABSTRACT :-

This report describes an inventory and stack control system implemented in java. The system models items in stack, supports common inventory operations (add, remove, search, update), and demonstrate a stack-based control feature for last-in first out (LIFO) handling of batches or temporary storage (Eg. returned items, staging area). The design emphalizes modularity, clear class responsibilities, error handling, and testable Condeon is included with explanations, sample test runs, and suggestions for future work.

2. INTRODUCTION :-

Inventory Control is essential for organization to track stock levels, Manage replenishment, and ensure smooth opperations. The stack Control Concept is Ueful when items are handle in a LIFO manner - for Ex; stocking pallets, temporary staging of returns, or handling expiry-prone goods when the most reacently added batch must be processed first. This project demonstrates a simple but estensible java implementation suitable for learning, allignments or as a foundations for GUI / DB - backed system.

Goals :-

* Provide basic inventory CRUD (Create, Read, Update, delete) operations.

* Demonstrate stack-based staging and batch processing.

* Produce clear, commented java code and a report describing design and testing.

3. System Design :-

3.1 Class Structure.

A class structure defines how different classes in the inventory stack control system are organized and connected Each class represents a specific component of the system with this own attributes (data) and methods (operations)

Main Classes Used

- Item Class - Represents individual inventory items with attributes such as item ID, name, Quality and price.
- Inventory Manager class - Manages a list of items and provides operations like adding, removing, updation, searching and displaying items.

Stack Class - Implements stack operations (POP, Push, display) used to track stack additions and removals.

Main Class - Contains the main method & provides a user interface to interact with the system.

Purpose of class Structure :-

* Organizes the system into modular and manageable Components.
* Promotes reusability and maintainability of Code.

3.2 Design Explanation

- **Simple Inventory Class**

→ Acts as the main controller or manager for the inventory system.

→ Stores product details in a HashMap<String, product>, where the Product ID is the key and the value is a product object.

→ Uses a scanner for Console-based user input.

→ Contains all system operations like adding, searching, viewing, updating and selling products.

→ Menu driven structures liked to make the program interactive and user-friendly.

- Product Inner Class -

→ Represent a single product.

→ Holds essential attributes such as Product ID, name, price and Quantity.

→ Uses to string() to format product information for display.

→ Simple data model to encapsulate product properties.

- Data Structure Choice

→ HashMap provides O(1) average time complexity for insert search, and update operations.

→ Efficient when accessing product records by ID.

- Input Validation

→ Method getIntInput() and getDoubleInput() ensure correct numeric input and prevent system crashes due to invalid input.

- Inventory Management operations

* Add product: Allows creation of new products with unique ID.

* Update Quantity: Changes stock level directly.

* Search product: Searches by Both ID & name.

* Sell product: Deducts sold Quantity and Calculates Sale Value.

* View All: Lists entire inventory and Calculates the total inventory value.

• User Interface :-

* Implemented using console text menu for easy interaction.

* Repeats menu until the user choose to exit.

4. Implementation :-

4.1 Code Snippets :-

```
import java.util.*;

class Product {
    private String id;
    private String name;
    private int quantity;
    private double price;

    Product(String id, String name, int quantity, double price) {
        this.id = id;
        this.name = name;
        this.quantity = quantity;
        this.price = price;
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public int getQuantity() { return quantity; }
    public double getPrice() { return price; }

    public void setQuantity(int quantity) { this.quantity = quantity; }
    public void setPrice(double price) { this.price = price; }

    public void display() {
        System.out.println("-----------------------------------------------");
        System.out.println("Product ID      : " + id);
        System.out.println("Product Name    : " + name);
        System.out.println("Quantity in Stock: " + quantity);
```

```
        System.out.println("Price per Unit   : ₹" + price);
        System.out.println("----------------------------------------------");
    }
}

public class Main {

    private static Scanner sc = new Scanner(System.in);
    private static HashMap<String, Product> inventory = new HashMap<>();

    public static void main(String[] args) {
        int choice;

        do {
            System.out.println("\n===== INVENTORY & STOCK CONTROL SYSTEM =====");
            System.out.println("1. Add Product");
            System.out.println("2. Update Stock");
            System.out.println("3. Delete Product");
            System.out.println("4. View All Products");
            System.out.println("5. Search Product");
            System.out.println("6. Exit");
            System.out.print("Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1: addProduct(); break;
                case 2: updateStock(); break;
                case 3: deleteProduct(); break;
```

```
            case 4: viewProducts(); break;
            case 5: searchProduct(); break;
            case 6: System.out.println("Exiting... Thank you!"); break;
            default: System.out.println("Invalid choice! Try again.");
        }
    } while (choice != 6);
}

private static void addProduct() {
    System.out.print("Enter Product ID: ");
    String id = sc.next();

    if (inventory.containsKey(id)) {
        System.out.println("Product already exists!");
        return;
    }

    System.out.print("Enter Product Name: ");
    String name = sc.next();

    System.out.print("Enter Quantity: ");
    int qty = sc.nextInt();

    System.out.print("Enter Price: ");
    double price = sc.nextDouble();

    Product p = new Product(id, name, qty, price);
    inventory.put(id, p);
```

```
    System.out.println("Product added successfully!");
}

private static void updateStock() {
    System.out.print("Enter Product ID to update: ");
    String id = sc.next();

    if (!inventory.containsKey(id)) {
        System.out.println("Product not found!");
        return;
    }

    Product p = inventory.get(id);

    System.out.print("Enter new quantity: ");
    int qty = sc.nextInt();
    p.setQuantity(qty);

    System.out.print("Enter new price: ");
    double price = sc.nextDouble();
    p.setPrice(price);

    System.out.println("Product updated successfully!");
}

private static void deleteProduct() {
    System.out.print("Enter Product ID to delete: ");
```

```
    String id = sc.next();

    if (inventory.remove(id) != null)
        System.out.println("Product deleted successfully!");
    else
        System.out.println("Product not found!");
}

private static void viewProducts() {
    if (inventory.isEmpty()) {
        System.out.println("No products to display!");
        return;
    }

    for (Product p : inventory.values()) {
        p.display();
    }
}

// ✔ Missing method added (Fixes your error)
private static void searchProduct() {
    System.out.print("Enter Product ID to search: ");
    String id = sc.next();

    if (inventory.containsKey(id)) {
        System.out.println("Product found:");
        inventory.get(id).display();
    } else {
```

4. IMPLEMENTATION :-

4.1 Code Snippets.

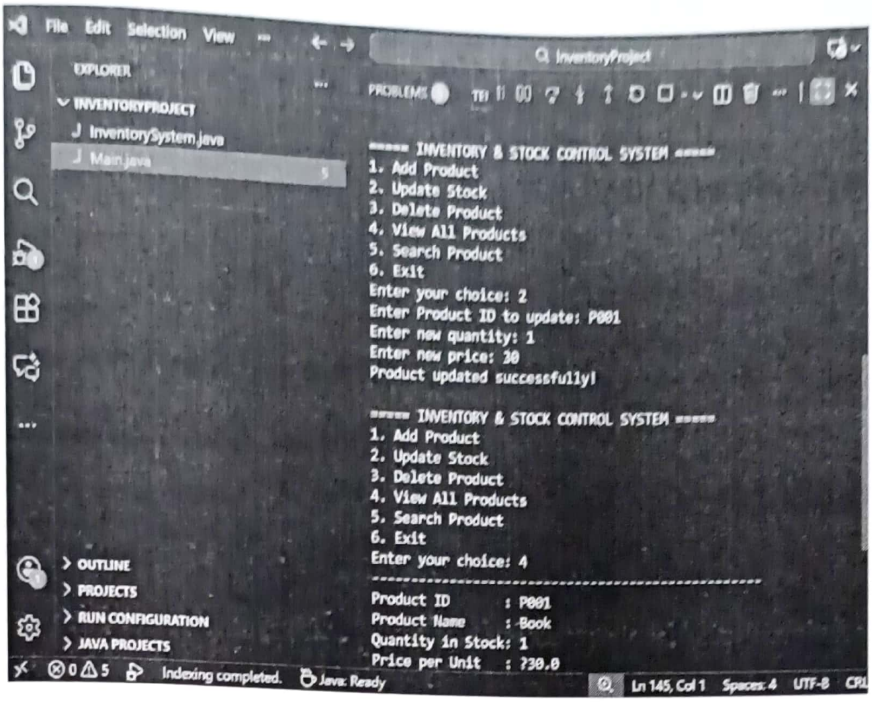
4.2 Explanation.

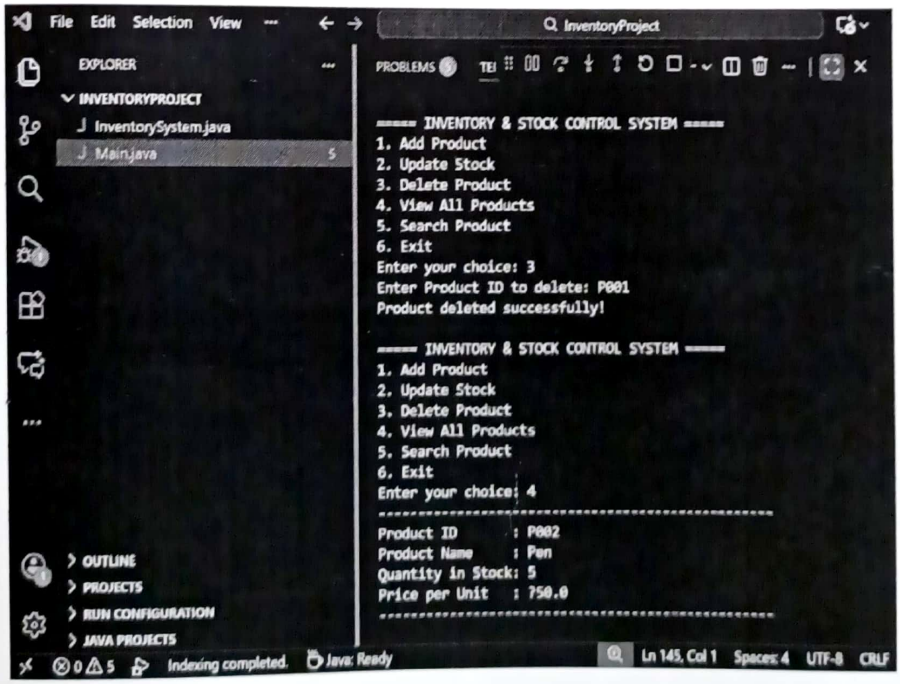
4.2 Implementation Explanation :-

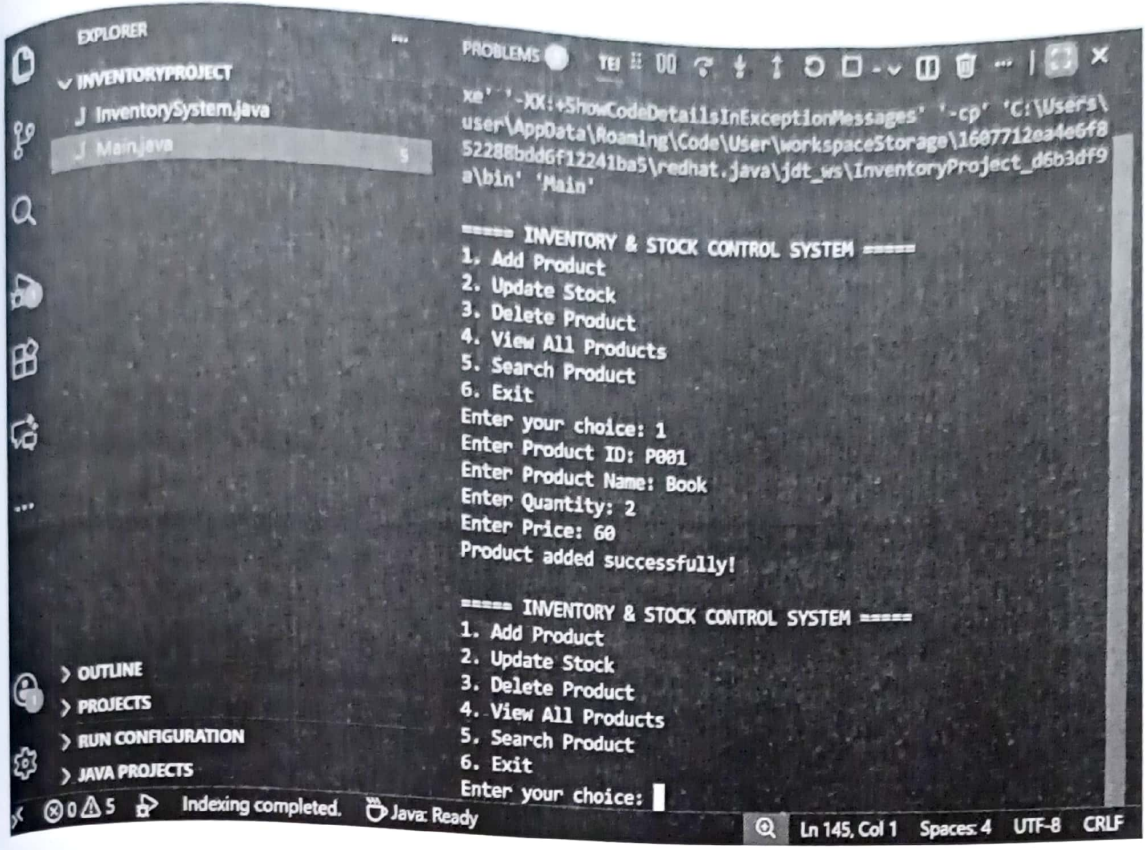
- The demo seeds the inventory with three items and pushes a small return onto the stack. The transfer TOP Inventory method either increases the exiting item's Quantity or adds the item if it doesn't exit.
- Inventory, remove Quantity attemps to subtract Quantity and returns false if insufficient stock; this keeps the inventory stat Consistent.
- The CLI has an optional interactive mode to try operations manually.

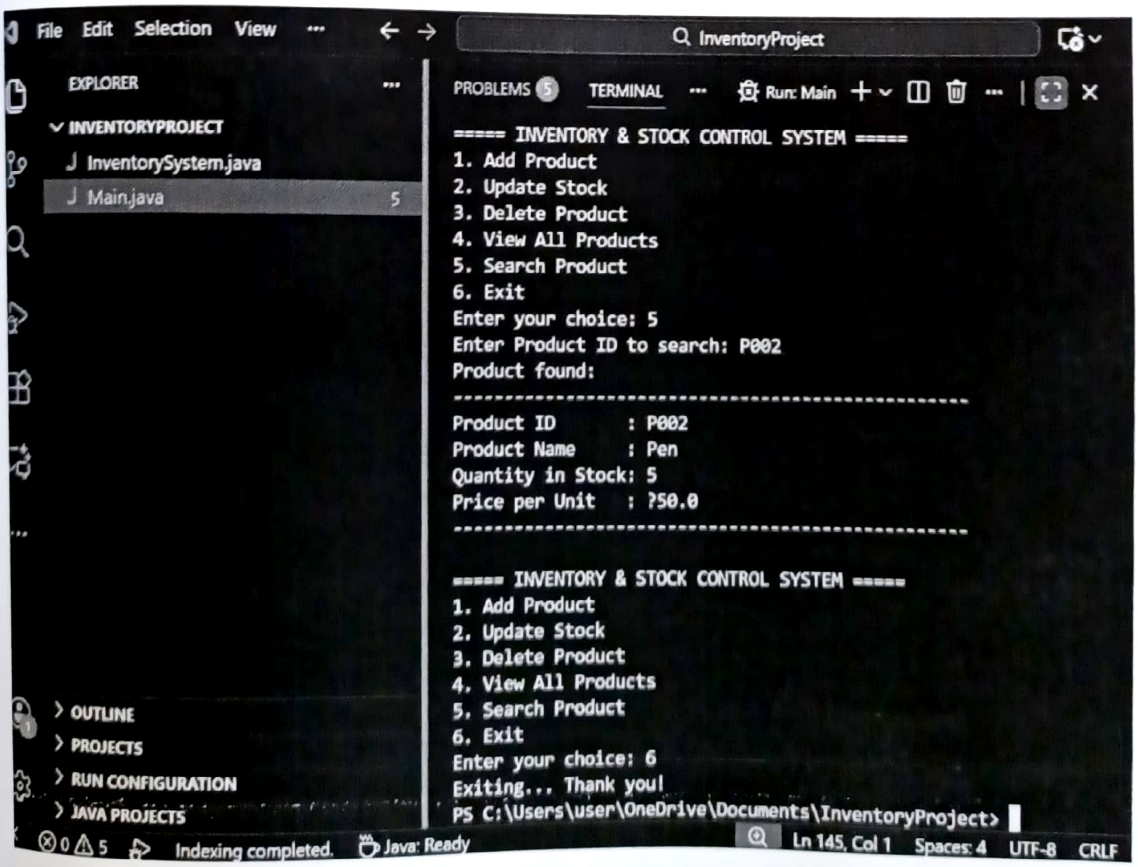
5. Testing and Output :-

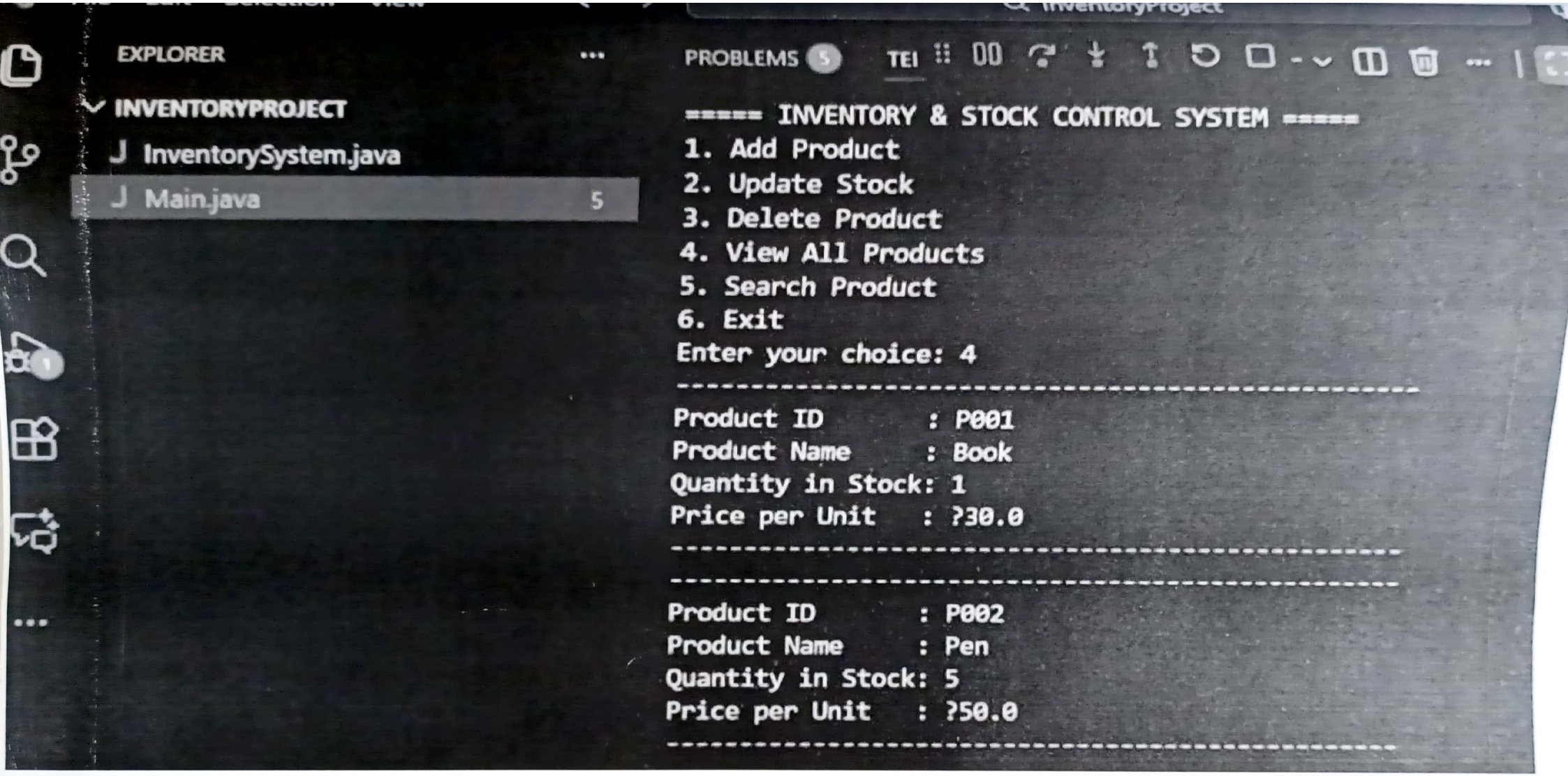
Output Snapshots :-











6. Conclusion :-

The Inventory and stock control system succesfully manages and tracks product stock using efficient object oriented programing principle. The system simplifies operations such as adding items, updating quantities, removing stock, and viewing inventory details. By using stack operations, it effectively records item movement in and out of the system. The project demonstrates how Java can be used to build a reliable and structured inventory control application that improves accuracy, reduces manual effort, and enhances stock visibility.

7. Future Scope :-

The System can be further improved with advanced features to enhance performance and real-world usability possible extension include:

- Integration with database systems like MySQL or oracle for persistent data storage.
- Implementation of Graphical user Interface (GUI) using JavaFX or Swing for better user interaction.
- Addition of barcode or QR Code scanning for faster item identification and billing.
- Real-time notifications and alters for low stock and expiry tracking.

Cloud-based access and mobile Application development for remote Inventory management.

Role-Based login System for admin and staff authorization.

GM University
Davanagere

GM UNIVERSITY
Innovating Minds

TOPIC : INVENTORY AND STACK CONTROL SYSTEM

Presented by:

Anjali M	U24E01DS007
Bhavana SS	U24E01DS011
Jeevitha BV	U24E01DS026
Nikitha SB	U24E01DS049

Presented to:

Mrs. Ashwini G T
Dept of Data science
Davanagere

TABLE OF CONTENTS

1. ABSTRACT

- Inventory management is essential for tracking products, materials, and stock levels efficiently.
- The project "Inventory and Stack Control System" helps store product details, manage item quantities, and avoid shortages or overstocking.
- The system allows adding, deleting, updating, and displaying items using stack operations for efficient management.
- It ensures accuracy, reduces manual errors, and supports real-time decision-making for inventory handling.

2.INTRODUCTION

•Inventory control involves monitoring stock availability and maintaining optimal levels for business operations.

•Traditional manual systems are slow and prone to mistakes.

•This computerized system automates inventory updates and manages product records effectively.

•Integrating stack data structure improves order-based processing (LIFO) such as tracking recently added stock.

3. SYSTEM DESIGN

- **Components**
- **Product Class** – stores product ID, name, price, and quantity.
- **Inventory Class** – acts as a container for stock using stack and list operation.
- **Stack Operations Implemented**
  - **Push:** Add product into stock
  - **Pop:** Remove last added product
  - **Peek:** View top item in stack
  - **Display All:** Display full inventory

4.SYSTEM DESIGN EXPLANATION

- Uses **HashMap / Array List / Stack** to store product objects.
- Provides menu-driven operations for easy user interaction.
- GUI/Console interface for retrieving and updating item information.
- Ensures fast access, efficient stock calculations, and safe transactions.

5.CONCLUSION

- ✓ The Inventory and Stack Control System provides an efficient way to track and manage stock.
- ✓ Replaces manual record-keeping with automated and error-free operations.
- ✓ Stack usage ensures systematic controlling and retrieving of inventory data.
- ✓ Helps organizations improve productivity and resource planning.

6. FUTURE SCOPE

- Integration with **barcode and QR-code scanner**.
- **Mobile application** with cloud-based data synchronization.
- Automatic **low-stock alerts via SMS/Email**.
- **AI-based forecasting** for demand prediction.
- Support for multiple warehouses and suppliers.
- Real-time dashboard using **IoT and web interface**.

GM UNIVERSIT
Innovating Mi

Thank you all