CERTIFICATE

This is to certify that [Your Name], a student of Class XII-[Section] at [Your School Name], has successfully completed the project titled **"----------"** under the supervision of [Project Supervisor Name] during the academic year [2025-26].

This project has been completed in fulfillment of the requirements for the Computer Science course and adheres to the guidelines issued by CBSE, New Delhi.

The project work meets the expected standards and has been evaluated by the undersigned.

Signature of Principal: _____

Signature of Supervisor: _____

Signature of External Examiner: _____

Date: _____

Place: _____

Student Details:

Student Names:

Class : XII

Subject: Information Technology

School Name: Torch Bearers Convent School

Affiliation By.: CBSE Board

ABSTRACT

The Inventory Management System project is a basic software solution designed to automate the management of product stock using core programming concepts such as classes, arrays, and loops. It provides users with the ability to perform CRUD (Create, Read, Update, Delete) operations efficiently on inventory items, enabling accurate tracking of stock levels, product details, and transactions. This system addresses the challenges of manual inventory management, such as errors, delays, and data loss, by ensuring reliable and timely updates of stock status. The project is designed to be user-friendly and requires minimal system resources, making it suitable for small to medium-scale operations. By automating inventory tasks, this system enhances operational efficiency and provides a foundation for scalable inventory controls.

This project was implemented by the team members Rohan, Anam, Khushi (Gill), and Bhumi, who contributed to coding, testing, and documentation processes.

Table of Contents

Introduction

Inventory management is a crucial aspect of any business that deals with goods and materials. It involves the process of ordering, storing, tracking, and controlling inventory to ensure that the right amount of stock is available at the right time. Proper inventory management helps companies reduce costs, prevent overstocking or stockouts, and improve overall operational efficiency.

This project focuses on developing a basic Inventory Management System using fundamental programming concepts such as classes, arrays, and loops. The system is designed to automate and simplify the management of product stock by supporting CRUD (Create, Read, Update, Delete) operations. It allows users to add new products, update existing stock levels, view inventory details, and delete obsolete items.

By implementing this system, users can effectively maintain accurate records of product stock, manage transactions smoothly, and minimize errors prevalent in manual inventory handling. This project serves as an essential tool for small businesses or beginner programmers to understand and apply object-oriented programming and data management techniques.

Objectives

The primary objectives of this Inventory Management System (Basic) project are:

- To develop a reliable and efficient system to automate the tracking and management of product stock using fundamental programming concepts such as classes, arrays, and loops.
- To enable seamless CRUD (Create, Read, Update, Delete) operations for managing product inventory including adding new products, updating stock quantities, deleting obsolete products, and viewing available stock.
- To minimize manual errors and increase accuracy in inventory records by implementing validation and transaction handling mechanisms.
- To facilitate easy retrieval and display of product information and stock levels through user-friendly interfaces and looping constructs.
- To improve inventory control and decision-making by maintaining up-to-date stock details that reflect sales and restocking activities.
- To provide a basic but scalable solution suitable for small businesses aiming to manage inventory efficiently without complex software dependencies.
- To enhance understanding of object-oriented programming through practical implementation of classes representing products and arrays for storing multiple items.

System Design and Classes

System Architecture Overview

The Inventory Management System is designed using object-oriented principles to manage product stock efficiently. The system uses a modular approach, dividing responsibilities among classes that represent products, stock management, and transactions. Data is stored in arrays to list multiple product entries, and operations such as adding, updating, and deleting product stock records are performed using loops and methods in classes.

Key Classes and Their Roles

- **Product Class:** Represents a product in the inventory. It includes attributes such as Product ID, Product Name, Quantity in Stock, and Price. Methods in this class handle adding new products, updating product details, and displaying product information.
- **Inventory Class:** Manages the collection of products stored in an array. It includes methods to add a product to the inventory array, remove a product, search for a product by ID or name, update stock quantity, and display the entire product list.
- **Transaction Class:** Handles stock transactions like sales and restocking. It ensures that stock quantities are adjusted properly and validates available stock before allowing deductions.

Class Attributes and Methods

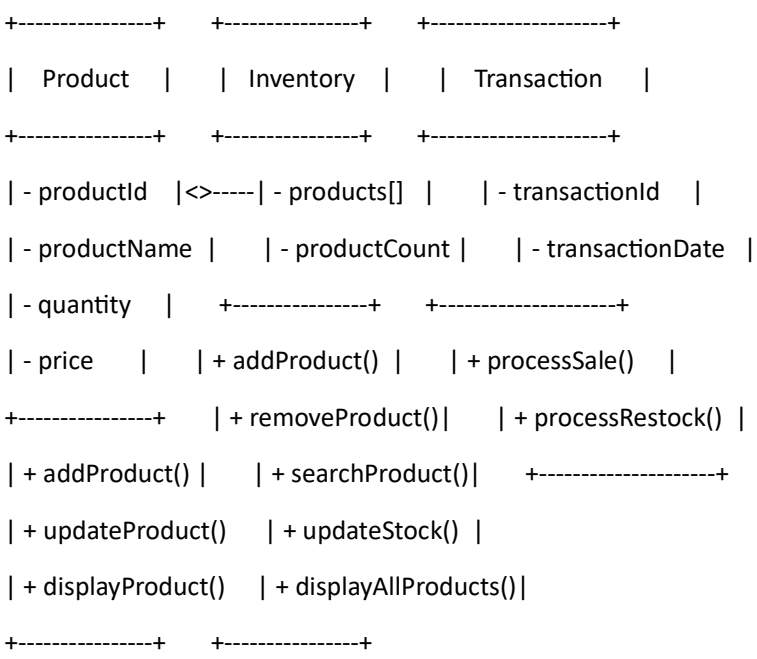
Class	Attributes	Methods
Product	int productId	addProduct()
	String productName	updateProduct()
	int quantity	displayProduct()
	double price	
Inventory	Product[] products	addProduct(Product p)
	int productCount	removeProduct(int productId)
		searchProduct(int productId)
		updateStock(int productId, int quantity)
		displayAllProducts()
Transaction	int transactionId	processSale(int productId, int quantity)
	Date transactionDate	processRestock(int productId, int quantity)

System Flow

- The system initializes with an empty Inventory array.
- Products are added to the inventory by instantiating Product objects and adding them to the Inventory array.
- Transactions for sales and restocking adjust the quantity of products.
- CRUD operations are performed by calling methods on Inventory and Product classes.

- Loops iterate through the array to display product lists and validate product existence.

UML Class Diagram (Simplified)



This simplified design ensures a clear separation of concerns and efficient management of inventory data using classes, arrays, and CRUD-like operations through loops and methods. It provides a solid foundation for implementing basic inventory functionalities.

Core Components and Code Snippets

- com.example.Product.Product.java

```java
package com.example.Product;

/**
 * Represents a product with id, name, and quantity.
 */
public class Product {
    private int id;
    private String name;
    private int quantity;

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

    // Getters and setters
    public int getId() { return id; }
    public String getName() { return name; }
    public int getQuantity() { return quantity; }
    public void setQuantity(int quantity) { this.quantity = quantity; }

    @Override
    public String toString() {
        return "Product ID: " + id + ", Name: " + name + ", Quantity: " +
quantity;
    }
}
```

- com.example.inventory.Inventory.java

```java
package com.example.Inventory;

import com.example.Product.Product;

import java.util.ArrayList;
import java.util.List;

/**
 * Manages a collection of products.
 */
public class Inventory {
    private List<Product> products;

    public Inventory() {
        products = new ArrayList<>();
    }

    // Add a product
    public void addProduct(Product product) {
        products.add(product);
        System.out.println("Product added: " + product.getName());
    }

    // Remove a product by ID
    public boolean removeProduct(int productId) {
        for (Product product : products) {
            if (product.getId() == productId) {
                products.remove(product);
```

```
                System.out.println("Removed product: " + product.getName());
                return true;
            }
        }
        System.out.println("Product not found: " + productId);
        return false;
    }

    // Update quantity for a product
    public boolean updateQuantity(int productId, int newQuantity) {
        for (Product product : products) {
            if (product.getId() == productId) {
                product.setQuantity(newQuantity);
                System.out.println("Updated quantity for " + product.getName() +
" to " + newQuantity);
                return true;
            }
        }
        System.out.println("Product not found: " + productId);
        return false;
    }

    // Display all products
    public void displayProducts() {
        if (products.isEmpty()) {
            System.out.println("Inventory is empty.");
            return;
        }
        System.out.println("Current Inventory:");
        for (Product product : products) {
            System.out.println(product);
        }
    }

    // Find product by ID
    public Product findProduct(int productId) {
        for (Product product : products) {
            if (product.getId() == productId) {
                return product;
            }
        }
        return null;
    }
}
```

-
- com.example.manager.InventoryManager.java

```
package com.example.Manager;

import com.example.Inventory.Inventory;
import com.example.Product.Product;

import java.util.Scanner;

/**
 * Provides user interaction for inventory management.
 */
public class InventoryManager {
    private Inventory inventory;
    private Scanner scanner;

    public InventoryManager() {
        inventory = new Inventory();
        scanner = new Scanner(System.in);
    }
```

```java
    public void runMenu() {
        while (true) {
            System.out.println("\nInventory Management System");
            System.out.println("1. Add Product");
            System.out.println("2. Remove Product");
            System.out.println("3. Update Product Quantity");
            System.out.println("4. Display Products");
            System.out.println("5. Search Product");
            System.out.println("6. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt();
            scanner.nextLine();  // consume newline

            switch (choice) {
                case 1 -> addProduct();
                case 2 -> removeProduct();
                case 3 -> updateQuantity();
                case 4 -> inventory.displayProducts();
                case 5 -> searchProduct();
                case 6 -> {
                    System.out.println("Exiting program.");
                    scanner.close();
                    return;
                }
                default -> System.out.println("Invalid option. Try again.");
            }
        }
    }

    private void addProduct() {
        System.out.print("Enter Product ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter Product Name: ");
        String name = scanner.nextLine();

        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();
        scanner.nextLine();

        Product product = new Product(id, name, quantity);
        inventory.addProduct(product);
    }

    private void removeProduct() {
        System.out.print("Enter Product ID to Remove: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        inventory.removeProduct(id);
    }

    private void updateQuantity() {
        System.out.print("Enter Product ID to Update: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter New Quantity: ");
        int quantity = scanner.nextInt();
        scanner.nextLine();

        inventory.updateQuantity(id, quantity);
    }

    private void searchProduct() {
```

```
            System.out.print("Enter Product ID to Search: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            Product product = inventory.findProduct(id);
            if (product != null) {
                System.out.println("Found Product:");
                System.out.println(product);
            } else {
                System.out.println("Product not found.");
            }
        }
    }
```

-

- InventoryManagementApp.java

```
import com.example.Manager.InventoryManager;

/**
 * Main class to start the com.example.inventory.Inventory Management System.
 */
public class InventoryManagementApp {
    public static void main(String[] args) {
        InventoryManager manager = new InventoryManager();
        manager.runMenu();
    }
}
```

-

Transaction Handling

Transaction handling in the system refers to managing all changes to the product stock, ensuring correct and consistent updates during operations such as adding new stock or processing sales.

- **Adding Stock:** When new products arrive or stock is replenished, the system increases the product's quantity using the addStock() method.
- **Removing Stock (Sales):** Before processing a sale, the system verifies whether sufficient stock is available. If yes, it deducts the quantity accordingly via the reduceStock() method.
- **Updating Stock:** For any modifications in inventory (corrections or adjustments), the system updates the quantity directly.
- **Validation:** The system ensures no negative quantities occur by checking stock availability before any reduction. It also validates input data for correctness.
- **Error Handling:** Invalid transactions such as reducing more stock than available trigger error messages and prevent the operation.
- **Loop Operations:** Arrays storing products are traversed using loops to locate the correct product based on ID for transaction processing.

This approach ensures reliable stock management, preventing inconsistencies and supporting audit trails in simple form. All transactions affect the in-memory product array and, optionally, a persistent file or database for durability.

Conclusion and Team Contribution

Conclusion

The Inventory Management System developed in this project effectively automates the process of tracking and managing product stock, replacing manual and error-prone methods with an efficient, accurate, and user-friendly application. Through the use of core programming concepts such as classes, arrays, and loops, combined with CRUD operations, the system allows seamless addition, deletion, updating, and viewing of product information. This automation not only improves inventory accuracy but also enhances operational efficiency by reducing human error and saving time. Although basic, the system lays a strong foundation for further enhancements, such as integrating database support, graphical user interfaces, or advanced transaction handling.

Team Contribution

- **Rohan:** Designed the system architecture and developed the Product class and CRUD operations.
- **Anam:** Implemented the array-based product storage, loops for managing stock transactions, and input validation.
- **Khushi (Gill):** Worked on the user interface components and tested various functions for correctness and reliability.
- **Bhumi:** Prepared the project documentation, including the report drafts, and ensured proper formatting and presentation.

Together, the team collaborated effectively to combine their strengths in programming and documentation, successfully delivering the project within the stipulated timeline.