**INVENTORY MANAGEMENT SYSTEM**

**A PROJECT REPORT**

***Submitted by***

**THULASIRAM BHEEMINENI**

**CHAITHANYA SULLURU**

**AMARNADH CHINTHAPOODHI**

**THATAMSETTY THRINETHRA**

**BALAJI VIJAYANAGARAM**

***of in partial fulfilment for the award of the***

***certification of Java***

***in***

**BACHELOR OF TECHNOLOGY**

**ARJUN COLLEGE OF TECHNOLOGY, COIMBATORE**

**INFOSYS FOUNDATION**

**FINSHING SCHOOL FOR EMPLOYABILTY**

**ICT ACADEMY**

SEPTEMBER 2024

**INFOSYS FOUNDATION**

**FINSHING SCHOOL FOR EMPLOYABILTY**

**ICT ACADEMY**

**BONAFIDE CERTIFICATE**

Certified that this project report **“Inventory Management System”** is the bonafide work of **“Thulasiram Bheemineni, Chaithanya Sulluru, Amarnadh Chinthapoodhi, Thatamsetty Thrinethra, Balaji Vijayanagaram”** who carried out the project work under my supervision.

Mrs. Ambika Veildurai., B.tech

**ICT ACADEMY**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE** |
| **1.** | **ABSTRACT** | **iv** |
| **2.** | **LIST OF TABLES** | **v** |
| **3.** | **LIST OF FIGURES** | **v** |
| **4.** | **LIST OF SYMBOLS** | **vi** |
| **5.** | **INTRODUCTION** | **1** |
| **6.** | **INVENTORY MANAGEMENT ARCHITECTURE** | **2** |
| **6.1** | **DATA INPUT LAYER** | **2** |
| **6.2** | **INVENTORY MANAGEMENT SOFTWARE** | **3** |
| **6.3** | **DATABASE LAYER** | **3** |
| **6.4** | **INTEGRATION LAYER** | **3** |
| **6.5** | **BUSINESS RULES AND LOGIC LAYER** | **4** |
| **6.6** | **USER INTERFACE LAYER** | **4** |
| **6.7** | **SECURITY AND ACCESS CONTROL** | **4** |
| **7.** | **ER DIAGRAM** | **5** |
| **8.** | **SOURCE CODE** | **6** |
| **9.** | **OUTPUT** | **11** |
| **10.** | **CONCLUSION** | **14** |

**1.ABSTRACT**

This project report details the development of a **Java-based Inventory Management System** designed to manage and track products, stock levels, and customer orders efficiently. The system is built using object-oriented programming principles, emphasizing modularity and scalability. It provides a comprehensive solution for adding and managing products, updating stock quantities, and maintaining accurate inventory records, ensuring that businesses can track product availability and handle stock updates effectively. The system’s architecture is centered around key classes, including Product, StockManagement, Order, AccountingIntegration, and Reporting, each responsible for a specific aspect of the inventory workflow.

In addition to inventory tracking, the system supports order management, allowing users to create and fulfill orders seamlessly. Integration with the accounting system enables automatic logging of sales transactions, offering businesses a streamlined process for maintaining financial records. Orders can be fulfilled through a simple and intuitive interface, ensuring efficient order processing and tracking of order status. This approach ensures accuracy and timeliness in fulfilling customer demands, avoiding stock shortages or excesses.

Furthermore, the system generates detailed reports on both stock levels and sales. The Reporting class provides features for creating sales reports based on customer orders and stock reports for reviewing current inventory levels. This enables businesses to make informed decisions regarding stock replenishment and sales strategies. Overall, the system demonstrates the practical application of Java programming concepts in solving real-world inventory management problems, enhancing operational efficiency for small and medium-sized enterprises.

**2.LIST OF TABLES**

1. Table 1: Product Information Database Schema
2. Table 2: Sales Information Database Schema
3. Table 3: Generating The Sales Report

**3.LIST OF FIGURES**

1. Figure 1: System Architecture Diagram
2. Figure 2: Entity-Relationship Diagram (ERD)
3. Figure 3: Product Main Class
4. Figure 4: Adding a Product Into The Cart
5. Figure 5: Updating The Order
6. Figure 6: Creating The Order (Not adding a product)
7. Figure 7: Creating The Order
8. Figure 8: Fulfilling The Order
9. Figure 9: Generating Sales Report

**4. LIST OF SYMBOLS**

1. DAO: Data Access Object
2. CRUD: Create, Read, Update, Delete
3. ORM: Object-Relational Mapping
4. MVC: Model-View-Controller
5. SQL: Structured Query Language
6. KPI: Key Performance Indicator
7. HTTP: Hypertext Transfer Protocol
8. REST: Representational State Transfer
9. API: Application Programming Interface
10. RDBMS: Relational Database Management System
11. UML: Unified Modeling Language
12. JSP: JavaServer Pages
13. JSON: JavaScript Object Notation
14. CSV: Comma-Separated Values

**5. INTRODUCTION**

Inventory management is a critical component of modern business operations, allowing companies to efficiently track and manage their stock of goods and materials. It involves monitoring the quantities of products available, determining reorder points, and optimizing storage practices to ensure that products are available when needed while avoiding overstocking. Effective inventory management is essential for balancing supply with customer demand, minimizing costs, and maximizing profitability. In industries like retail, manufacturing, and logistics, well-managed inventories can lead to significant competitive advantages.

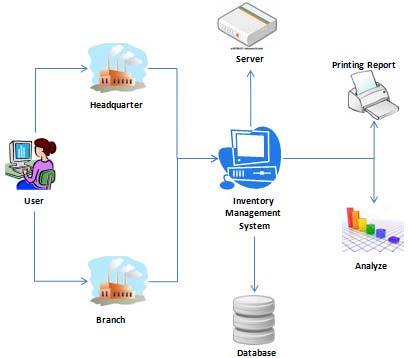
The importance of inventory management cannot be overstated in today’s fast-paced and competitive business environment. Companies that can maintain the right inventory levels can meet customer demands promptly and avoid missed sales opportunities. On the other hand, those that mismanage inventory may face issues such as excess stock, which leads to higher storage costs, or stock shortages, which result in dissatisfied customers. A good inventory management system helps companies balance these opposing risks, providing visibility into inventory levels in real time and ensuring operational efficiency.

Technological advancements have revolutionized inventory management, with automated systems and software tools now widely used to track inventory. These systems often integrate with other business functions, such as sales, accounting, and supply chain management, to provide a comprehensive overview of inventory performance. By utilizing inventory management software, businesses can automate stock updates, generate reports, and set alerts for reordering, which reduces manual effort and human error. In addition, data analytics tools help businesses forecast demand and optimize stock levels.

One of the key strategies in inventory management is determining the right reorder point and maintaining a just-in-time (JIT) inventory system. JIT focuses on reducing the amount of inventory on hand by receiving goods only when they are needed for production or sale. This minimizes holding costs and reduces the risk of obsolete inventory. However, it requires precise coordination between suppliers and the business to ensure timely delivery and avoid stockouts. For businesses with volatile demand, safety stock is often kept to manage unexpected spikes in customer orders.

In conclusion, effective inventory management is the foundation of smooth business operations. It directly influences customer satisfaction, cost efficiency, and profitability. By using modern technologies and optimizing inventory practices, companies can gain better control over their stock, meet demand effectively, and maintain a streamlined supply chain. In a global market where customer expectations are higher than ever, companies with strong inventory management practices can respond more agilely and maintain their competitive edge.

**6.INVENTORY MANAGEMENT ARCHITECTURE**



Inventory management architecture refers to the framework and components that enable the efficient tracking, storing, managing, and controlling of inventory in a business. This architecture is typically designed to ensure optimal stock levels, reduce costs, and improve the overall supply chain process. Modern inventory management architecture consists of various layers, including hardware, software, databases, and integrations with other business systems. Below is a breakdown of its core components:

**6.1. Data Input Layer:**

The data input layer is responsible for collecting and entering inventory data into the system. This layer includes tools such as barcode scanners, RFID (Radio Frequency Identification) tags, and manual data entry. Every time a product enters or leaves the warehouse, it is scanned or manually logged into the system to ensure real-time data accuracy. The key technologies involved in this layer are:

-**Barcode/RFID scanners:** Automatically capture product data such as identification numbers, quantities, and movement.

**-Mobile devices and applications:** Allow warehouse staff or inventory managers to input data remotely or on-site.

**- IoT (Internet of Things) devices:** Enable real-time tracking of inventory using sensors.

**6.2.** **Inventory Management Software:**

The software layer is the backbone of the inventory management system and typically includes modules for tracking stock levels, orders, sales, and deliveries. It can also predict future inventory needs based on historical data. Some of the functionalities of inventory management software include:

-**Product catalog management:** Allows businesses to manage product information such as product ID, name, price, and description.

**-Stock control and updates:** Automatically updates stock levels when new products are received or shipped.

-**Order management:** Helps track customer orders, fulfill orders, and manage returns.

-**Reporting and analytics:** Provides insights into sales trends, stock movement, and product performance.

6.**3. Database Layer:**

The database layer is where all inventory data is stored and managed. This includes information about products, stock levels, locations, suppliers, and customer orders. The database must be well-structured to allow for efficient queries, updates, and retrieval of data. A robust database ensures that the system remains scalable and can handle increasing volumes of transactions as the business grows. The common database technologies used are:

-**Relational databases (e.g., MySQL, PostgreSQL):** Store structured data in tables and provide relationships between various inventory records.

|  |  |  |  |
| --- | --- | --- | --- |
| Product Id | Product Name | Product Price | Stock Quantity |
| 142 | Realme | 150$ | 5 |
| 165 | Iphone 11 | 200$ | 6 |
| 2004 | HP I5 | 190$ | 15 |

Table 1: list of products ordered

|  |  |  |  |
| --- | --- | --- | --- |
| Product Id | Product Name | Updated Quantity | Price |
| 142 | Realme | 1 | 150$ |
| 165 | Iphone 11 | 2 | 200$ |

Table 2: Order Information

|  |  |  |  |
| --- | --- | --- | --- |
| Product Id | Order Id | Price | Order Ststus |
| 142 | 123645 | 150$ | Fulfilled |
| 165 | 1364924 | 200$ | Not Found |

Table 3: Order Status Information

**6.4. Integration Layer:**

Inventory management systems often need to integrate with other business systems to ensure smooth operations. This layer allows the system to communicate with accounting, sales, procurement, and customer relationship management (CRM) systems. Key integrations include:

**-ERP (Enterprise Resource Planning) integration:** Connects the inventory system with broader business processes, such as purchasing, accounting, and human resources.

**-Accounting software:** Automates the flow of financial data from inventory transactions to accounting, ensuring accurate cost tracking and revenue reporting.

**- CRM integration:** Connects customer data with inventory data, helping businesses manage customer orders and inventory needs more efficiently.

**6.5. Business Rules and Logic Layer:**

This layer contains the rules and logic that govern how the inventory system functions. For instance, it defines when the system should reorder stock, how to manage safety stock, and how to fulfill customer orders based on inventory levels. It also includes demand forecasting algorithms that predict when and how much stock will be needed based on historical trends. The business rules are essential for automating processes like:

**-Reorder point calculation:** Determines the minimum stock level at which new stock should be ordered.

**-Order fulfillment logic:** Determines the process for selecting inventory for customer orders and shipping it.

6.**6. User Interface Layer:**

The user interface (UI) layer provides an accessible interface for users, such as warehouse managers, inventory clerks, and decision-makers, to interact with the system. It allows users to perform various tasks, such as viewing stock levels, generating reports, and managing orders. The UI should be intuitive, allowing for easy navigation through features and ensuring smooth operation even by non-technical staff. Modern inventory management systems often have:

**-Desktop interfaces:** For detailed inventory management and reporting.

-**Mobile interfaces:** For on-the-go stock management by warehouse staff or field agents.

**-Dashboard views:** For quick insights and monitoring of key performance indicators (KPIs).

6.**7. Security and Access Control:**

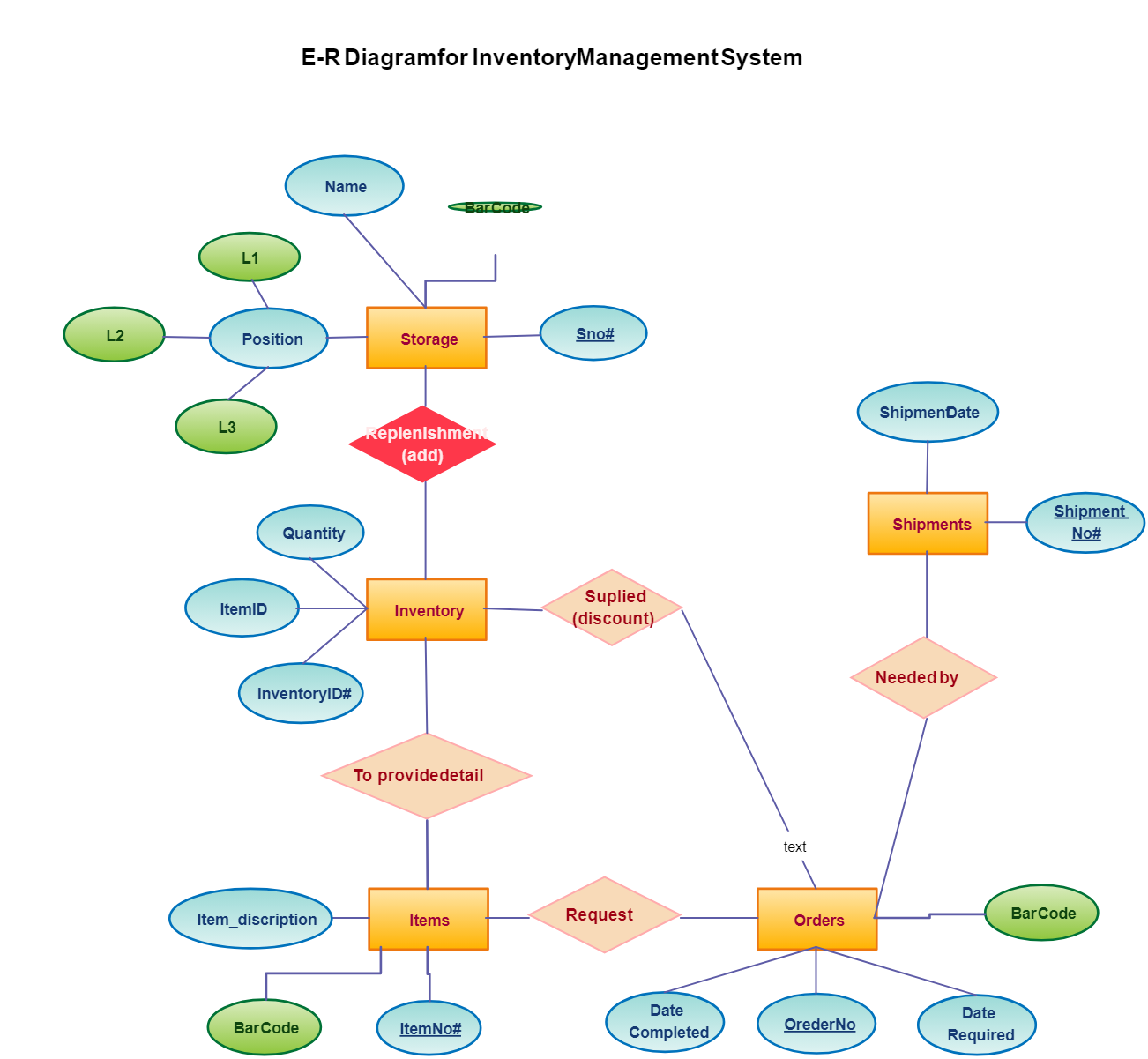
Inventory management architecture must ensure data integrity and security. Access to the system is often controlled based on user roles, ensuring that only authorized personnel can view or modify certain information. Key security features include:

**-User authentication and role-based access:** Ensures that sensitive inventory data is accessible only to authorized personnel.

**-Data encryption:** Protects inventory data, especially when transmitted between devices and systems.

**- Audit logs:** Track changes to inventory records, ensuring accountability and preventing fraud.

**7.ENTITY-RELATIONSHIP DIAGRAM:**

****

**8.SOURCE CODE:**

import java.util.\*;

// Product class

class Product {

private String productId;

private String productName;

private double price;

private int stockQuantity;

public Product(String productId, String productName, double price, int stockQuantity) {

this.productId = productId;

this.productName = productName;

this.price = price;

this.stockQuantity = stockQuantity;

}

public String getProductId() { return productId; }

public String getProductName() { return productName; }

public double getPrice() { return price; }

public int getStockQuantity() { return stockQuantity; }

public void updateStock(int quantity) {

this.stockQuantity += quantity;

}

public void updatePrice(double newPrice) {

this.price = newPrice;

}

@Override

public String toString() {

return productId + ": " + productName + " - " + stockQuantity + " units @ $" + price;

}

}

// Stock Management class

class StockManagement {

private Map<String, Product> productCatalog = new HashMap<>();

public void addProduct(Product product) {

productCatalog.put(product.getProductId(), product);

}

public void updateStock(String productId, int quantity) {

Product product = productCatalog.get(productId);

if (product != null) {

product.updateStock(quantity);

System.out.println("Stock updated: " + product);

} else {

System.out.println("Product not found.");

}

}

public Product getProduct(String productId) {

return productCatalog.get(productId);

}

public void displayAllProducts() {

for (Product product : productCatalog.values()) {

System.out.println(product);

}

}

}

// Order Management class

class Order {

private String orderId;

private List<Product> products;

private boolean fulfilled;

public Order(String orderId, List<Product> products) {

this.orderId = orderId;

this.products = new ArrayList<>(products);

this.fulfilled = false;

}

public String getOrderId() {

return orderId;

}

public void fulfillOrder() {

if (!fulfilled) {

fulfilled = true;

System.out.println("Order " + orderId + " has been fulfilled.");

} else {

System.out.println("Order " + orderId + " is already fulfilled.");

}

}

@Override

public String toString() {

return "Order ID: " + orderId + " - Fulfilled: " + fulfilled;

}

}

// Integration with Accounting System

class AccountingIntegration {

public void recordSale(String productId, int quantity, double price) {

System.out.println("Sale recorded for Product ID: " + productId + ", Quantity: " + quantity + ", Total: $" + (quantity \* price));

}

}

// Reporting and Analytics

class Reporting {

public void generateSalesReport(List<Order> orders) {

System.out.println("Sales Report:");

for (Order order : orders) {

System.out.println(order);

}

}

public void generateStockReport(StockManagement stockManagement) {

System.out.println("Stock Report:");

stockManagement.displayAllProducts();

}

}

// Main Inventory Management System

public class InventoryManagementSystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

StockManagement stockManagement = new StockManagement();

List<Order> orders = new ArrayList<>();

AccountingIntegration accounting = new AccountingIntegration();

Reporting reporting = new Reporting();

while (true) {

System.out.println("\nInventory Management System Menu:");

System.out.println("1. Add Product");

System.out.println("2. Update Stock");

System.out.println("3. Create Order");

System.out.println("4. Fulfill Order");

System.out.println("5. Generate Sales Report");

System.out.println("6. Generate Stock Report");

System.out.println("7. Exit");

System.out.print("Select an option: ");

int option = scanner.nextInt();

scanner.nextLine(); // Consume the newline

switch (option) {

case 1:

// Add Product

System.out.print("Enter Product ID: ");

String productId = scanner.nextLine();

System.out.print("Enter Product Name: ");

String productName = scanner.nextLine();

System.out.print("Enter Product Price: ");

double price = scanner.nextDouble();

System.out.print("Enter Stock Quantity: ");

int quantity = scanner.nextInt();

Product newProduct = new Product(productId, productName, price, quantity);

stockManagement.addProduct(newProduct);

System.out.println("Product added: " + newProduct);

break;

case 2:

// Update Stock

System.out.print("Enter Product ID: ");

String stockProductId = scanner.nextLine();

System.out.print("Enter quantity to add: ");

int stockQuantity = scanner.nextInt();

stockManagement.updateStock(stockProductId, stockQuantity);

break;

case 3:

// Create Order

System.out.print("Enter Order ID: ");

String orderId = scanner.nextLine();

System.out.print("Enter Product ID for the order: ");

String orderProductId = scanner.nextLine();

Product orderProduct = stockManagement.getProduct(orderProductId);

if (orderProduct != null) {

List<Product> orderProducts = new ArrayList<>();

orderProducts.add(orderProduct);

Order newOrder = new Order(orderId, orderProducts);

orders.add(newOrder);

accounting.recordSale(orderProductId, 1, orderProduct.getPrice());

System.out.println("Order created: " + newOrder);

} else {

System.out.println("Product not found.");

}

break;

case 4:

// Fulfill Order

System.out.print("Enter Order ID to fulfill: ");

String fulfillOrderId = scanner.nextLine();

boolean orderFound = false;

for (Order order : orders) {

if (order.getOrderId().equals(fulfillOrderId)) {

order.fulfillOrder();

orderFound = true;

break;

}

}

if (!orderFound) {

System.out.println("Order not found.");

}

break;

case 5:

// Generate Sales Report

reporting.generateSalesReport(orders);

break;

case 6:

// Generate Stock Report

reporting.generateStockReport(stockManagement);

break;

case 7:

// Exit

System.out.println("Exiting the system.");

scanner.close();

return;

default:

System.out.println("Invalid option. Try again.");

}

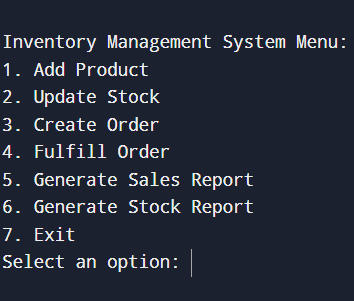
}

}

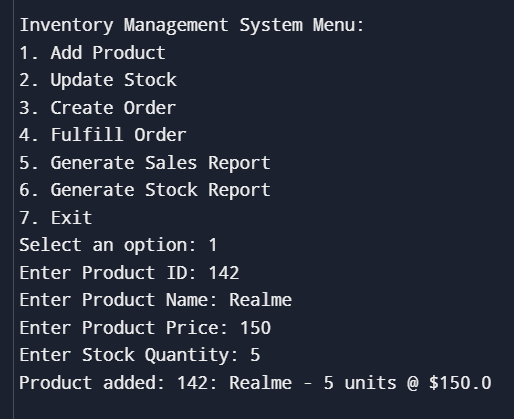
}

**9.OUTPUT:**

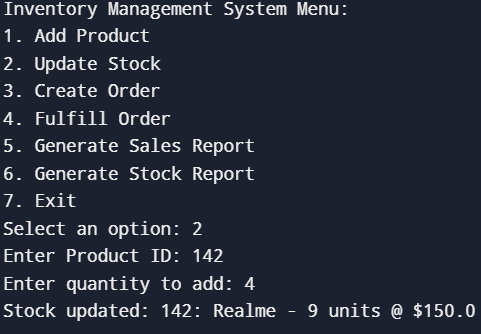
**MAIN CLASS:**



**PRODUCT CLASS:**

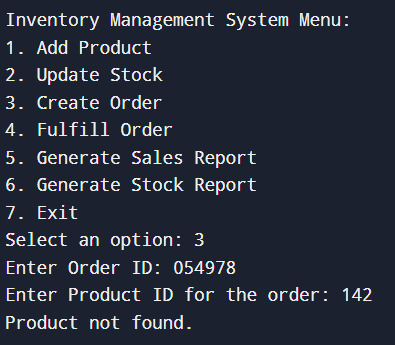
****

**UPDATING THE STOCK:**

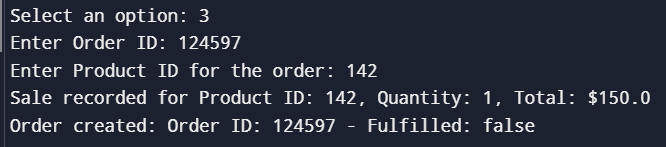
****

**CREATING THE ORDER:**

**IF THE PRODUCT IS NOT ADDED:**

****

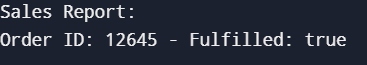
**IF THE PRODUCT IS ADDED:**

****

**FULFILLING THE ORDER:**

****

**GENERATING SALES REPORT:**

****

**10.CONCLUSION:**

The Inventory Management System is a well-structured program designed to manage products, orders, and stock levels within a business environment. It enables users to add products, update stock quantities, and create and fulfill orders efficiently. The system includes a Product class to manage individual product attributes like ID, name, price, and stock quantity, and a StockManagement class that handles the addition of products and updates to stock levels. Additionally, orders can be created and linked to products, which are processed using the Order class, allowing the system to track whether an order has been fulfilled.

The system is integrated with an accounting feature through the AccountingIntegration class, recording product sales when orders are created, which simulates real-world sales tracking. It also has a reporting module (Reporting class) to generate both sales and stock reports, providing insights into the performance of the inventory system. Through its comprehensive menu-based system, users can easily manage the entire inventory lifecycle, from product addition to order fulfilment, while ensuring seamless data recording and reporting. The use of clear object-oriented design makes the system modular, extensible, and easy to maintain.