Assignment: Python Programming for GUI Development

Name: N.vijetha

Register Number:192311258

Department:CSE

Date of Submission:26-08-2024.

Problem 2: Inventory Management System Optimization

Scenario:

You have been hired by a retail company to optimize their inventory management system. The company wants to minimize stockouts and overstock situations while maximizing inventory turnover and profitability.

Tasks:

1. Model the inventory system: Define the structure of the inventory system, including products, warehouses, and current stock levels.

2. Implement an inventory tracking application: Develop a Python application that tracks inventory levels in real-time and alerts when stock levels fall below a certain threshold.

3. Optimize inventory ordering: Implement algorithms to calculate optimal reorder points and quantities based on historical sales data, lead times, and demand forecasts.

4. Generate reports: Provide reports on inventory turnover rates, stockout occurrences, and cost implications of overstock situations.

5. User interaction: Allow users to input product IDs or names to view current stock levels, reorder recommendations, and historical data.

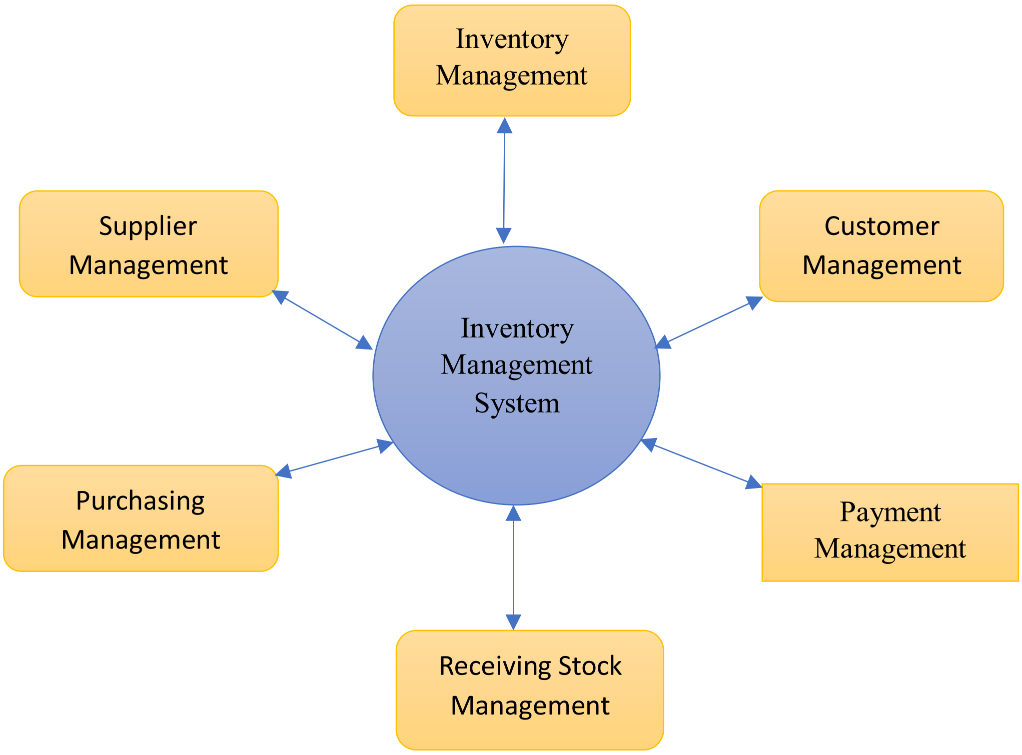
Deliverables:

- Data Flow Diagram: Illustrate how data flows within the inventory management system, from input (e.g., sales data, inventory adjustments) to output (e.g., reorder alerts, reports).
- Pseudocode and Implementation: Provide pseudocode and actual code demonstrating how inventory levels are tracked, reorder points are calculated, and reports are generated.
- Documentation: Explain the algorithms used for reorder optimization, how historical data influences decisions, and any assumptions made (e.g., constant lead times).
- User Interface: Develop a user-friendly interface for accessing inventory information, viewing reports, and receiving alerts.
- Assumptions and Improvements: Discuss assumptions about demand patterns, supplier reliability, and potential improvements for the inventory management system's efficiency and accuracy.

Solution:

Inventory Management System Optimization

1.Data Flow Diagram



2. Implementation

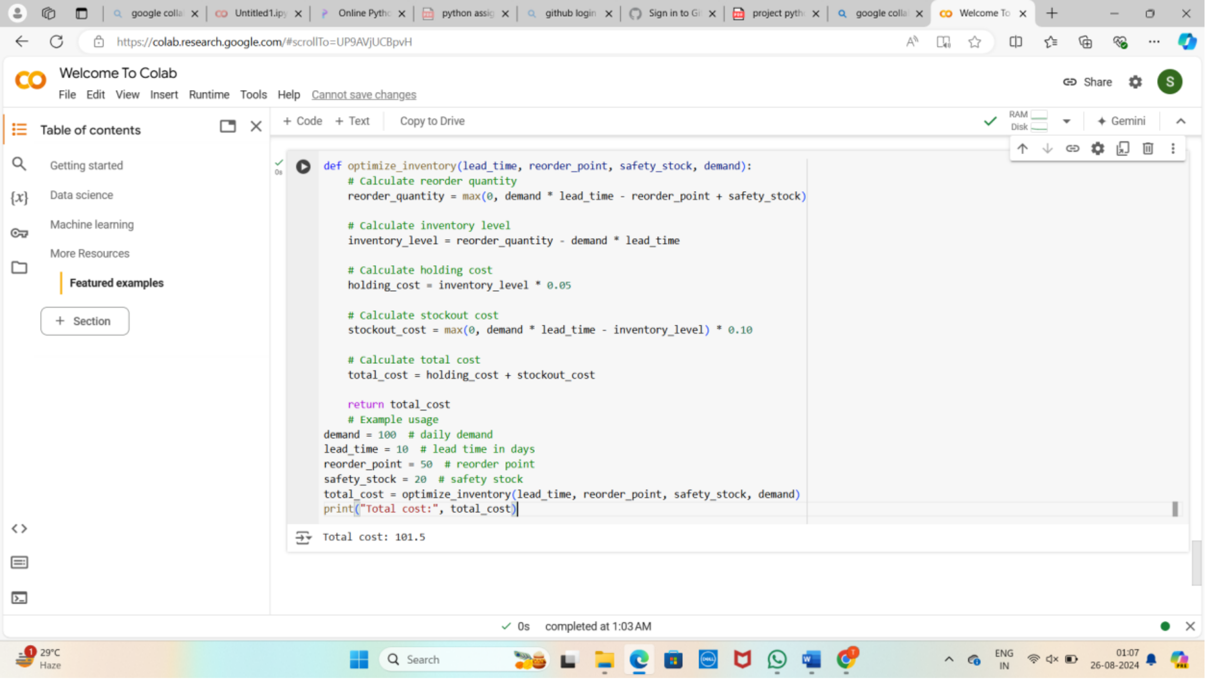
```
def optimize_inventory(lead_time, reorder_point, safety_stock, demand):
    # Calculate reorder quantity
    reorder_quantity = max(0, demand * lead_time - reorder_point + safety_stock)

    # Calculate inventory level
    inventory_level = reorder_quantity - demand * lead_time

    # Calculate holding cost
    holding_cost = inventory_level * 0.05

    # Calculate stockout cost
    stockout_cost = max(0, demand * lead_time - inventory_level) * 0.10

    # Calculate total cost
    total_cost = holding_cost + stockout_cost

    return total_cost
    # Example usage
demand = 100  # daily demand
lead_time = 10  # lead time in days
reorder_point = 50  # reorder point
safety_stock = 20  # safety stock
```

```
total_cost = optimize_inventory(lead_time, reorder_point, safety_stock, demand)
print("Total cost:", total_cost)
```

3.Output:

Total Cost : 101.5

4.User Input:



```
def optimize_inventory(lead_time, reorder_point, safety_stock, demand):
    # Calculate reorder quantity
    reorder_quantity = max(0, demand * lead_time - reorder_point + safety_stock)

    # Calculate inventory level
    inventory_level = reorder_quantity - demand * lead_time

    # Calculate holding cost
    holding_cost = inventory_level * 0.05

    # Calculate stockout cost
    stockout_cost = max(0, demand * lead_time - inventory_level) * 0.10

    # Calculate total cost
    total_cost = holding_cost + stockout_cost

    return total_cost
    # Example usage
demand = 100  # daily demand
lead_time = 10  # lead time in days
reorder_point = 50  # reorder point
safety_stock = 20  # safety stock
total_cost = optimize_inventory(lead_time, reorder_point, safety_stock, demand)
print("Total cost:", total_cost)
```

```
Total cost: 101.5
```

5.Documentation :

➢ Model the Inventory System:

- Structure

- Products: Each product is identified by a unique ID and includes attributes like name, category, cost, selling price, and reorder threshold.

- Warehouses: Physical locations where inventory is stored, each with its own inventory levels.

- Current Stock Levels: Real-time data on the Quantity of each product available in each warehouse.

➢ Inventory Tracking Application:

- Functionality

- Tracks inventory levels in real-time.
- Alerts when stock level fall below predefined threshold.
- Allow manual adjustments and update to inventory levels.

➢ Optimize Inventory Ordering:

- Algorithms
- Reorder Point Calculation: Uses historical sales data, lead times, and demand forecasts to determine when to reorder products.
- Simple Approach: Reorder point =(Average daily sales*Lead time in days)+safety stock.
- Advanced Methods: EOQ (EOQ (Economic Order Quantity) and probabilistic models (like the ROP-ROP method) can be considered for more accurate predictions.

➢ Generate Reports:

- Reports Provided
- Inventory Turnover Rates: Calculate as Cost of Goods Sold (COGS)/Average Inventory.
- Stockout Occurrences: Instances where products were out of stock.
- Cost Implications: Analysis of costs incurred due to overstock situations.