Report On

# Inventory Management

Submitted in partial fulfillment of the requirements of the Course project in

Semester IV of Second Year Computer Engineering

by

Aniket Chavan (Roll No. 26)

Gaurav Desale (Roll No. 38)

Chinmay Devrukhkar (Roll No. 40)

Supervisor

Prof. Sneha Mhatre

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**



**(2023-24)**

**Vidyavardhini's College of Engineering & Technology**

**Department of Computer Engineering**

**CERTIFICATE**

This is to certify that the project entitled **“Inventory Management**” is a bonafide work of "Aniket Chavan (Roll No. 26), Gaurav Desale (Roll No. 38), Chinmay Devrukhkar (Roll No. 40)" submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in SBL-Python semester IV of Second Year Computer Engineering.

**Supervisor**

|  |  |  |
| --- | --- | --- |
| Prof. Name Surname |  |  |
| Internal Examiner |  | External Examiner |
| Dr Megha Trivedi  Head of Department |  | Dr. H.V. Vankudre  Principal |

**Abstract**

Efficient inventory management is crucial for the success of any organization, impacting operational costs, customer satisfaction, and overall profitability. This abstract presents a comprehensive approach to inventory management aimed at optimizing resource utilization and enhancing supply chain performance. The proposed framework integrates advanced technologies, such as RFID (Radio Frequency Identification), IoT (Internet of Things), and ), to automate and streamline various inventory processes.

The key components of this approach include demand forecasting, inventory classification, replenishment strategies, and performance measurement. Leveraging historical data and predictive analytics, organizations can accurately forecast demand, reducing stockouts and overstock situations. Inventory classification techniques, such as ABC analysis, prioritize items based on their contribution to revenue and ensure efficient allocation of resources. Moreover, dynamic replenishment strategies, including Just-in-Time (JIT) and Vendor Managed Inventory (VMI), optimize inventory levels while minimizing holding costs and obsolescence risks.

Furthermore, real-time monitoring and analysis facilitated by IoT devices enable proactive inventory management, allowing organizations to respond swiftly to fluctuations in demand and supply. AI-driven algorithms optimize order quantities, lead times, and safety stock levels, adapting to changing market conditions and enhancing responsiveness. Additionally, performance metrics, such as inventory turnover and fill rates, provide insights into the effectiveness of inventory management strategies, facilitating continuous improvement initiatives.

By adopting this holistic approach to inventory management, organizations can achieve operational excellence, reduce carrying costs, and enhance customer satisfaction through improved product availability. The proposed framework serves as a roadmap for organizations seeking to optimize their inventory management practices and gain a competitive edge in today's dynamic business environment.

**Contents**

**Pg. No**

|  |  |
| --- | --- |
| 1 Problem statement | 1 |
| 1.1 Block diagram, description and Working | 2 |
| 1.2 Module Description | 3 |
| 1.3 Description of Software & Hardware components | 4 |
| 2 Experimental Code | 5 |
| 2.1 Result and Conclusion | 6 |
| 3 References | 7 |

**Problem statement**

The problem statement for inventory management typically revolves around efficiently managing the flow of goods or products in and out of a company's inventory. It involves balancing the need to have enough stock to meet customer demand while minimizing excess inventory that ties up capital and storage space. Key aspects of the problem statement include:

1. Optimal Stock Levels: Determining the optimal levels of inventory to minimize stockouts (running out of products) while avoiding overstocking (having too much inventory).

2. Demand Forecasting: Accurately forecasting customer demand to ensure that enough stock is available to fulfill orders without excessive surplus.

3. Inventory Tracking: Implementing systems to track inventory in real-time, enabling businesses to know exactly what they have on hand, where it is located, and when it needs to be replenished.

4. Ordering and Replenishmen: Establishing efficient processes for ordering new inventory or replenishing stock, considering lead times, supplier reliability, and economic order quantity (EOQ).

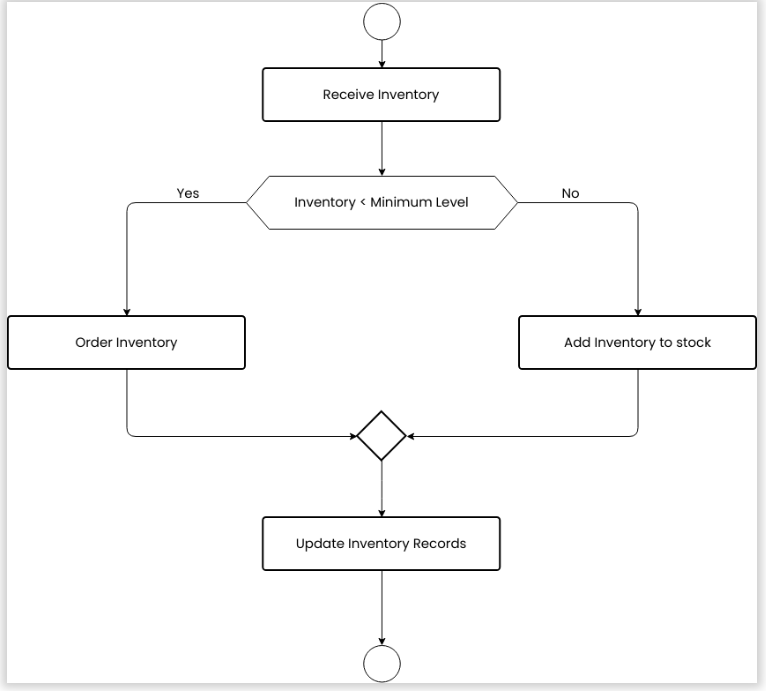
5. Inventory Classification: Classifying inventory items based on their importance, demand variability, and other factors to apply appropriate management strategies (e.g., ABC analysis).

6. Inventory Cost Management: Managing costs associated with holding inventory, including storage costs, carrying costs (e.g., insurance, taxes), and the cost of obsolescence or spoilage.

7. Supplier Management: Establishing relationships with reliable suppliers and managing supplier performance to ensure timely delivery of goods and favorable terms.

8. Inventory Optimization: Implementing strategies such as just-in-time (JIT) inventory, economic order quantity (EOQ), or vendor-managed inventory (VMI) to optimize inventory levels and reduce costs.

**Block diagram**

****

**Module Description :**

A module description for an inventory management system might include the following components:

1. Introduction: Briefly introduce the purpose and scope of the module, highlighting its importance in efficiently managing inventory within an organization.

2. Features: Outline the key features and functionalities provided by the module. This may include inventory tracking, demand forecasting, order management, reporting and analytics, supplier management, etc.

3. User Interface: Describe the user interface elements and how users interact with the module. This could include screens for viewing current inventory levels, placing orders, generating reports, etc.

4. Inventory Tracking: Explain how the module enables real-time tracking of inventory levels across different locations or warehouses. Detail the methods used for barcode scanning, RFID technology, or manual input to update inventory records.

5. Demand Forecasting: Discuss how the module supports demand forecasting to predict future inventory needs based on historical data, sales trends, and other factors. Explain any algorithms or methodologies used for demand prediction.

6 Order Management: Describe how the module facilitates order processing, including the ability to create purchase orders, track order status, and manage supplier relationships. Discuss any automated reorder triggers or alerts for low inventory levels.

7. Reporting and Analytics: Explain the reporting capabilities of the module, such as generating inventory turnover reports, ABC analysis, stockout reports, etc. Discuss how users can gain insights from these reports to make informed decisions.

8. Integration: Discuss any integration capabilities with other systems such as accounting software, ERP systems, or e-commerce platforms. Explain how data flows between systems to ensure accuracy and consistency.

**Description of software & hardware used and its programming :**

* **Software Used:** Python, Tkinter, MySQL
* **Hardware Requirements:** Standard computer system with Python environment installed.
* **Programming Languages:** Python for application development, SQL for database operations.

**Code :**

import sqlite3

import tkinter as tk

from tkinter import ttk, messagebox

# Connect to the SQLite database

conn = sqlite3.connect('inventory.db')

cursor = conn.cursor()

# Create table if not exists

cursor.execute('''CREATE TABLE IF NOT EXISTS products (

                    id INTEGER PRIMARY KEY,

                    name TEXT NOT NULL,

                    quantity INTEGER NOT NULL

                )''')

conn.commit()

# Define functions for adding, updating, and viewing products

def add\_product():

    name = name\_entry.get()

    quantity = quantity\_entry.get()

    try:

        quantity = int(quantity)

        cursor.execute('''INSERT INTO products (name, quantity) VALUES (?, ?)''', (name, quantity))

        conn.commit()

        messagebox.showinfo("Success", "Product added successfully.")

        view\_inventory()

    except ValueError:

        messagebox.showerror("Error", "Please enter a valid quantity.")

def update\_product\_quantity():

    product\_id = id\_entry.get()

    new\_quantity = new\_quantity\_entry.get()

    try:

        product\_id = int(product\_id)

        new\_quantity = int(new\_quantity)

        cursor.execute('''UPDATE products SET quantity = ? WHERE id = ?''', (new\_quantity, product\_id))

        conn.commit()

        messagebox.showinfo("Success", "Product quantity updated successfully.")

        view\_inventory()

    except ValueError:

        messagebox.showerror("Error", "Please enter valid product ID and quantity.")

def view\_inventory():

    inventory\_window = tk.Toplevel(root)

    inventory\_window.title("Current Inventory")

    inventory\_window.geometry("400x300")

    # Create a Treeview widget

    tree = ttk.Treeview(inventory\_window, columns=("ID", "Name", "Quantity"), show="headings")

    tree.heading("ID", text="ID")

    tree.heading("Name", text="Name")

    tree.heading("Quantity", text="Quantity")

    tree.pack(expand=True, fill="both")

    cursor.execute('''SELECT \* FROM products''')

    rows = cursor.fetchall()

    for row in rows:

        tree.insert("", "end", values=row)

# Main window

root = tk.Tk()

root.title("Inventory Management")

root.geometry("400x400")

add\_frame = tk.Frame(root)

add\_frame.pack(pady=10, anchor='w')

name\_label = tk.Label(add\_frame, text="Product Name:")

name\_label.grid(row=0, column=0, padx=5, pady=5, sticky='w')

name\_entry = tk.Entry(add\_frame)

name\_entry.grid(row=0, column=1, padx=5, pady=5, sticky='w')

quantity\_label = tk.Label(add\_frame, text="Quantity:")

quantity\_label.grid(row=1, column=0, padx=5, pady=5, sticky='w')

quantity\_entry = tk.Entry(add\_frame)

quantity\_entry.grid(row=1, column=1, padx=5, pady=5, sticky='w')

add\_button = tk.Button(add\_frame, text="Add Product", command=add\_product)

add\_button.grid(row=2, column=0, columnspan=2, pady=5, sticky='w')

update\_frame = tk.Frame(root)

update\_frame.pack(pady=10, anchor='w')

id\_label = tk.Label(update\_frame, text="Product ID:")

id\_label.grid(row=0, column=0, padx=5, pady=5, sticky='w')

id\_entry = tk.Entry(update\_frame)

id\_entry.grid(row=0, column=1, padx=5, pady=5, sticky='w')

new\_quantity\_label = tk.Label(update\_frame, text="New Quantity:")

new\_quantity\_label.grid(row=1, column=0, padx=5, pady=5, sticky='w')

new\_quantity\_entry = tk.Entry(update\_frame)

new\_quantity\_entry.grid(row=1, column=1, padx=5, pady=5, sticky='w')

update\_button = tk.Button(update\_frame, text="Update Quantity", command=update\_product\_quantity)

update\_button.grid(row=2, column=0, columnspan=2, pady=5, sticky='w')

view\_button = tk.Button(root, text="View Inventory", command=view\_inventory)

view\_button.pack(pady=10, anchor='w')

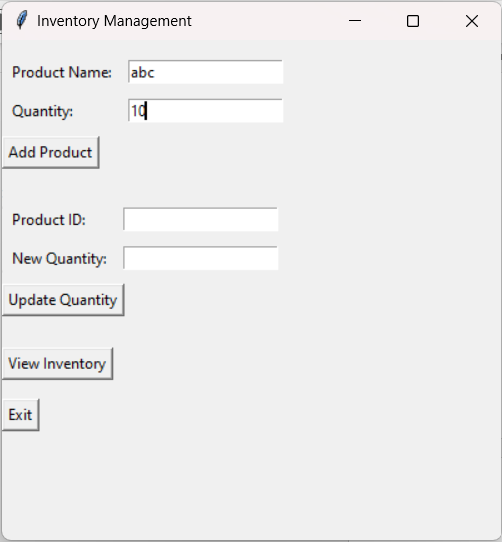
exit\_button = tk.Button(root, text="Exit", command=root.quit)

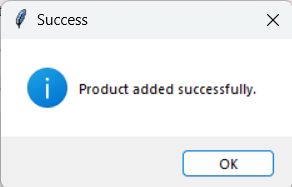
exit\_button.pack(pady=5, anchor='w')

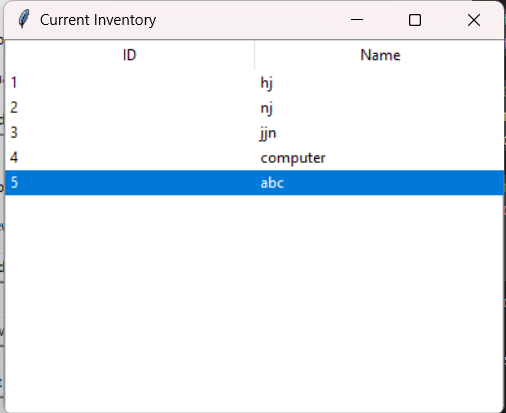
root.mainloop()

# Close connection

conn.close()







**Results and Conclusion :**

Improved Efficiency and Productivity: The event management module proved to be instrumental in improving the efficiency and productivity of event planning teams by automating repetitive tasks, streamlining processes, and centralizing information. This allowed teams to focus more on strategic planning and creative aspects of event management.

Better Decision Making: The availability of real-time data and analytics within the module empowered event organizers to make data-driven decisions and quickly respond to changing requirements or unforeseen challenges. This resulted in more informed decision-making and better outcomes for events.

Positive Impact on Event Success: Overall, the event management module had a positive impact on the success of events by ensuring smoother planning and execution, enhanced attendee satisfaction, and effective risk management. It played a crucial role in delivering memorable and successful events that met or exceeded stakeholders' expectations.

Recommendations for Improvement: While the event management module demonstrated significant benefits, there may be areas for improvement such as enhancing mobile accessibility, integrating with additional third-party tools, or incorporating more advanced analytics capabilities. Continuous feedback from users and stakeholders will be valuable for further refining and enhancing the module's functionalities.