**Inventory Management System**

**Problem Statement**

A large-scale manufacturing company requires a Python-based automated inventory management system to streamline the tracking, procurement, and usage of raw materials and finished goods. The system must integrate with various production units, warehouse systems, and supplier networks to ensure optimal stock levels, minimize production delays, and avoid overstocking or stockouts.

The system should:

* Monitor the real-time availability of materials.
* Automate reordering when inventory falls below a predefined threshold.
* Manage the transfer of raw materials between warehouses.
* Generate reports on inventory turnover and forecast future demand based on historical data.
* Send alerts for discrepancies or delays in shipments.
* Support multiple concurrent processes to handle inventory updates from various production units simultaneously.
* Ensure data security across all departments.

**Project Description**

This Python-based system is designed to provide seamless inventory management for a large manufacturing company. By utilizing object-oriented programming (OOP) principles, the system manages the core operations of inventory tracking, reordering, material transfers, and forecasting. It ensures that the inventory system can be accessed and updated concurrently from various production units while safeguarding sensitive data via an authentication mechanism.

The system is divided into multiple modules:

* **Inventory Management**: Track the availability and stock levels of materials.
* **Reordering System**: Automatically triggers a reorder when stock levels fall below a predefined threshold.
* **Material Transfer System**: Handles material transfers between warehouses and raises alerts for delays.
* **Forecasting System**: Generates reports based on historical data to forecast future demand and calculate inventory turnover.
* **Alert System**: Sends alerts for shipment delays or stock discrepancies via SMTP-based email notifications.
* **Security**: Ensures authentication for secure access to inventory data.

**Operations**

1. **Inventory Management**
   * **Add Material**: Allows the addition of new materials to the inventory.
   * **Update Stock**: Updates the stock level of an existing material.
   * **Display Stock**: Displays current stock levels for all materials.
2. **Reorder System**
   * **Check for Reorder**: Checks stock levels and triggers an automatic reorder for materials with stock levels below a threshold.
   * **Reorder Alert**: If no materials need reordering, the system outputs a message indicating no reorder is required.
3. **Transfer System**
   * **Transfer Materials**: Manages the transfer of materials between warehouses, including delays.
   * **Shipment Delay Alert**: Sends email alerts when delays exceed a specified duration using SMTP.
4. **Forecasting**
   * **Inventory Turnover Report**: Calculates the inventory turnover rate for each item.
   * **Sales Forecast**: Generates a forecast based on historical sales data with an assumed growth rate.
5. **Security**
   * **Authentication**: Ensures that only authorized users can access the system by verifying their credentials.

**Sample Input and Output**

**1.Authentication**

**Input:**

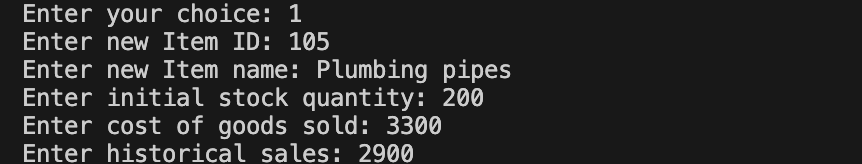
****

**Output:**

****

**2. Adding a Material**

**Input**:



**Output**:

****

**3. Displaying Stock Levels**

**Output**:

A black screen with white text

Description automatically generated

**3. Transfer Materials**

**Input**:

A black background with white text

Description automatically generated

**Output**:



**4. Inventory Turnover Report**

**Output**:

A screen shot of a report

Description automatically generated

**5. Forecast Report**

**Output**:

A black screen with white text

Description automatically generated

**Technologies & Concepts Learned During Training**

**1. SMTP (Simple Mail Transfer Protocol)**

SMTP is a protocol used to send emails from one server to another. In this project, I used SMTP for the **Alert System** to send automated email notifications for shipment delays. I learned how to configure an SMTP server, create an email message using MIME Text, and send it securely with TLS.

**2. OOP (Object-Oriented Programming) Concepts**

Throughout this project, I applied OOP principles such as:

* **Classes and Objects**: To model various system components (e.g., Inventory, Reorder, Transfer, Forecasting).
* **Encapsulation**: Each class encapsulates its functionality, making the code modular and maintainable.
* **Inheritance**: Although not used explicitly in this project, the concept of inheritance could be extended to enhance the code by creating a base class for shared functionality.

**3. Pandas**

I used the pandas library to manage and process the inventory data in a tabular format. Pandas is highly efficient in handling large datasets, and it allowed me to:

* **Generate Reports**: By transforming inventory data into DataFrame objects and calculating metrics like inventory turnover and sales forecasts.

**4. Authentication**

The **Security** class implemented a basic authentication system, where user credentials are stored in a dictionary and checked upon login. This ensures that only authorized personnel can perform sensitive operations like updating stock levels and managing materials.

**5. Threading**

Python’s threading module allows multiple threads to run concurrently, which is useful in scenarios where multiple users need to interact with the system at the same time.

**Required Modules**

To run the project, the following Python modules need to be installed:

1. pandas - For data manipulation and analysis.
2. smtplib - For sending email notifications.
3. email.mime - For constructing email messages.

**Conclusion**

This inventory management system project has been a comprehensive application of Python programming concepts. It helped me solidify my understanding of OOP, SMTP for sending email alerts, and working with the pandas library for data analysis.