Inventory Management System for a Retail Store

-Name:B.Harika

Regd no. :23331A0712

CSIT(2nd year)

Problem statement:

Create a program to manage inventory for a small retail store.

Introduction:

The **Inventory Management System** (IMS) is a simple command-line application designed to manage products in an inventory. It allows users to perform basic inventory operations such as adding new products, updating existing products, viewing product details, and generating low-stock reports. The system helps store product information such as price, stock, and category, and is intended for small-scale inventory tracking and management.

The system provides a user-friendly interface where users can:

1. Add new products (name, price, stock, category).
2. Update product stock or price.
3. View all products in a tabular format.
4. Generate a list of low-stock items based on a threshold.

Implementation:

```
# Inventory Management System
def add_product(products, name, price, stock, category):
    if name in products:
        print( f"Error: Product '{name}' already exists.")
    products[name] = {
        'price': price,
        'stock': stock,
        'category': category
    }
    print(f"Product '{name}' added successfully.")

def update_product(products, name, new_price=None, new_stock=None):
    if name not in products:
        print( f"Error: Product '{name}' does not exist.")
    if new_price is not None:
        products[name]['price'] = new_price
    if new_stock is not None:
        products[name]['stock'] = new_stock
    print(f"Product '{name}' updated successfully.")

def view_products(products):
    if not products:
        print( "No products available.")
    header =f"{'Name':<15}{'Price':<10}{'Stock':<10}{'Category':<15}"
    lines = [header, "=" * len(header)]
    for name, details in products.items():
        lines.append(f"{name:<15}{details['price']:<10}{details['stock']:<10}{details['category'
]:<15}")
    print("\n".join(lines))


def low_stock_report(products, threshold):
    low_stock_items = [
        print(f"{name} (Stock: {details['stock']})")
        for name, details in products.items()
        if details['stock'] < threshold
    ]
    if not low_stock_items:
        print("No low-stock items found.")
    print("Warning: " + ", ".join(low_stock_items))


# Main Program
products = {}
while True:
    print("\nInventory Management System")
    print("1. Add Product")
    print("2. Update Product")
    print("3. View Products")
    print("4. Low-Stock Report")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == "1":
        name = input("Enter product name: ")
        price = float(input("Enter product price: "))
        stock = int(input("Enter product stock: "))
        category = input("Enter product category: ")
        print(add_product(products, name, price, stock, category))

    elif choice == "2":
        name = input("Enter product name to update: ")
        update_choice = input("Update (1) Price or (2) Stock? ")
        if update_choice == "1":
            new_price = float(input("Enter new price: "))
            print(update_product(products, name, new_price=new_price))
        elif update_choice == "2":
            new_stock = int(input("Enter new stock: "))
            print(update_product(products, name, new_stock=new_stock))
        else:
            print("Invalid choice.")

    elif choice == "3":
        print(view_products(products))

    elif choice == "4":
        threshold = int(input("Enter low-stock threshold: "))
        print(low_stock_report(products, threshold))

    elif choice == "5":
        print("Exiting program. Goodbye!")
        break

    else:
        print("Invalid choice. Please try again.")
```

Explanation:

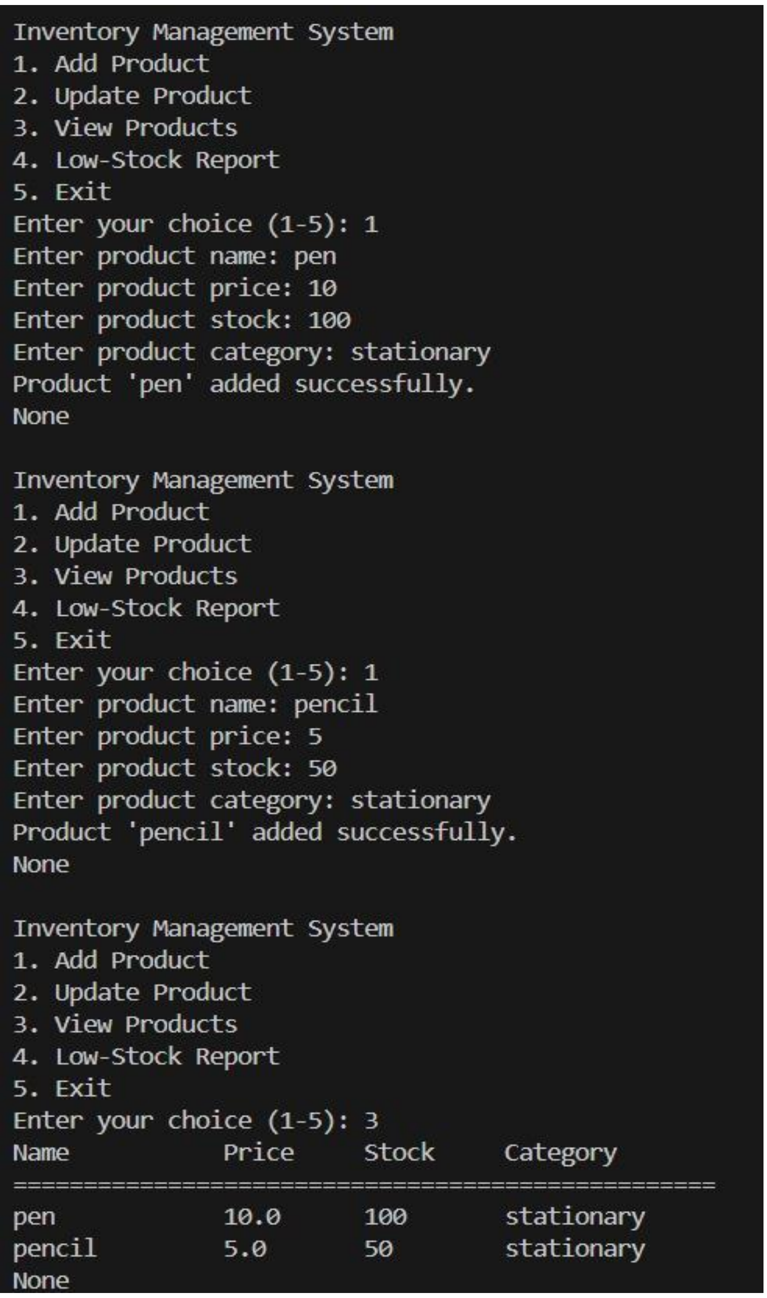
Key Functions:

1. **add_product(products, name, price, stock, category)**
   - Adds a new product to the inventory if it doesn't already exist.
   - **Parameters**: products (inventory dictionary), name, price, stock, and category.
   - **Returns**: Success or error message based on product existence.
2. **update_product(products, name, new_price=None, new_stock=None)**
   - Updates the price and/or stock of an existing product.
   - **Parameters**: products (inventory dictionary), name, new_price, and new_stock.
   - **Returns**: Success or error message if the product doesn't exist.
3. **view_products(products)**
   - Displays all products in the inventory in a formatted table.
   - **Parameters**: products (inventory dictionary).
   - **Returns**: A table displaying product name, price, stock, and category. If empty, shows "No products available."
4. **low_stock_report(products, threshold)**
   - Generates a report of products with stock levels below a specified threshold.
   - **Parameters**: products (inventory dictionary), threshold (stock threshold).
   - **Returns**: A list of low-stock items or a message if none are found.

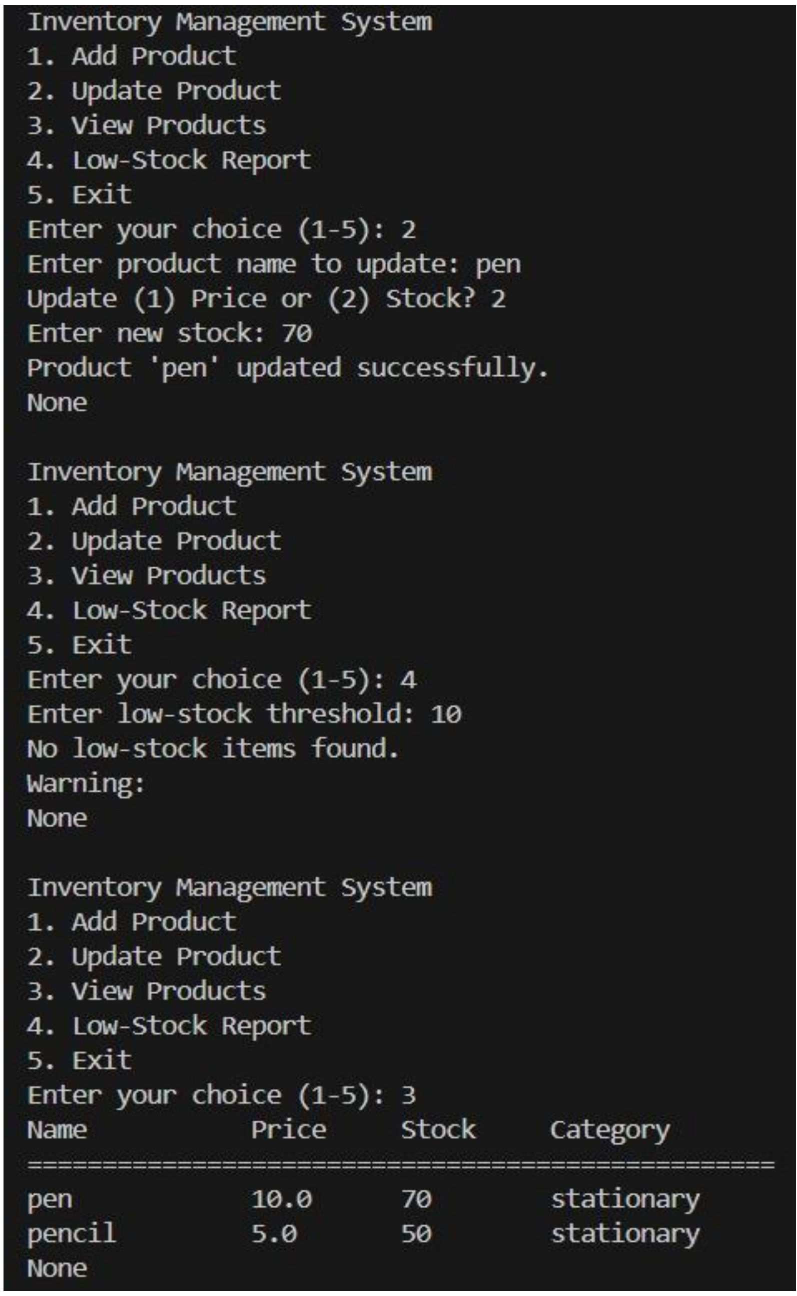
Main Program Workflow:

1. The program runs in a loop and displays a menu with five options:
   - Add Product
   - Update Product
   - View Products
   - Low-Stock Report
   - Exit
2. Based on the user's input, the corresponding function is called:
   - **Option 1**: Adds a new product by calling add_product.
   - **Option 2**: Updates a product's price or stock using update_product.
   - **Option 3**: Displays the inventory using view_products.
   - **Option 4**: Displays products with low stock using low_stock_report.
   - **Option 5**: Exits the program.
3. The loop continues until the user selects option 5 (Exit).

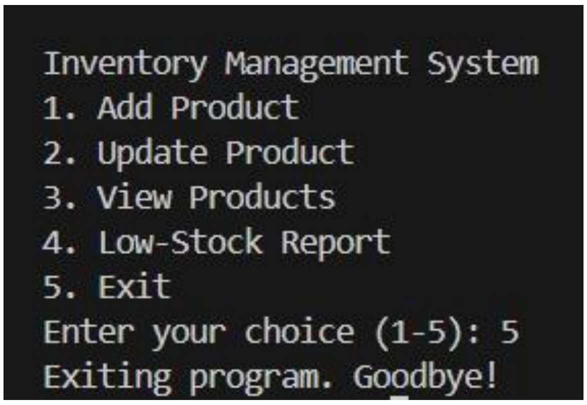
Result:



```
Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 1
Enter product name: pen
Enter product price: 10
Enter product stock: 100
Enter product category: stationary
Product 'pen' added successfully.
None

Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 1
Enter product name: pencil
Enter product price: 5
Enter product stock: 50
Enter product category: stationary
Product 'pencil' added successfully.
None

Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 3
Name           Price     Stock     Category
==================================================
pen            10.0      100       stationary
pencil         5.0       50        stationary
None
```



```
Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 2
Enter product name to update: pen
Update (1) Price or (2) Stock? 2
Enter new stock: 70
Product 'pen' updated successfully.
None

Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 4
Enter low-stock threshold: 10
No low-stock items found.
Warning:
None

Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 3
Name           Price     Stock     Category
==================================================
pen            10.0      70        stationary
pencil         5.0       50        stationary
None
```



```
Inventory Management System
1. Add Product
2. Update Product
3. View Products
4. Low-Stock Report
5. Exit
Enter your choice (1-5): 5
Exiting program. Goodbye!
```

Here are the key points learned from this project:

This code demonstrates key Python concepts, including using dictionaries for data storage, string formatting for display, and list comprehensions for filtering low-stock items. It also covers input validation, control flow with conditionals, and looping. These concepts are crucial for building efficient, user-friendly inventory management systems.

conclusion:

The **Inventory Management System** is a Python application for managing product inventories. It allows users to add, update, view, and generate low-stock reports for products. The system ensures no duplicate products are added, validates user input, and displays product details in a structured format. It also generates low-stock alerts based on a user-defined threshold. With a simple text-based menu, the system provides an easy-to-use solution for efficiently managing small to medium inventories.