Advanced Inventory Management System with Invoicing

Git repository

Table of Contents

Name: Raja Kanwar

Overview

This project is a class-based Python application designed to manage inventory, track sales and returns, and generate invoices in PDF format. It includes features for viewing, adding, updating, and removing products, as well as recording sales and returns transactions. A command-line interface (CLI) is provided for user interaction.

Objective

Develop a class-based Python application for managing inventory, including features for tracking sales, returns, and generating invoices in PDF format.

Features

- Class Design:

  Design classes for Product, Inventory, Transaction, Sale, Returns, and Invoice.

  Utilize appropriate inheritance and encapsulation techniques to ensure modularity and reusability.

- Inventory Management:

  View Products: Implement a method to display all available products along with their quantities.

  Manage Products: Provide APIs to add, update, and remove products from the inventory.

  Track Details: Maintain product quantities, prices, and categories.

- Sales Returns Tracking:

  Record Sales: Capture sales transactions including product ID, quantity sold, sale price, and transaction date.

Record Returns: Log returns including product ID, quantity returned, reason for return, and return date.

Update Inventory: Adjust inventory quantities based on sales and returns.

- Invoice Generation:

  Generate PDF Invoices: Create invoices in PDF format for sales transactions.

  Include Details: Each invoice should contain product names, quantities, prices, total amount, and transaction date.

  Query Invoices: Provide functionality to list invoices for specific products or customers.

- User Interaction:

  CLI Interface: Command-line interface for interacting with the system.

  - Features: Commands for adding products, viewing products, recording sales, recording returns, and generating PDF invoices.

System Requirements

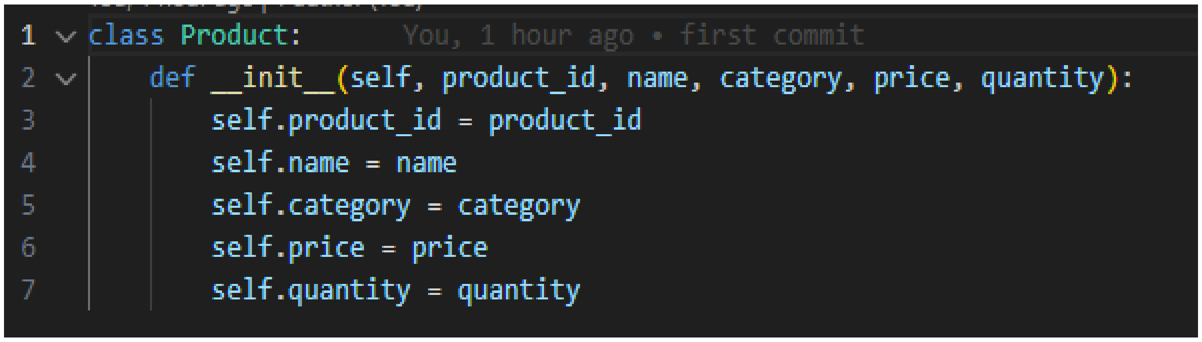
- Python 3.7 or higher
- `FPDF` library for generating PDFs (`pip install fpdf`)

System Components With Screen Shot

- Product Class : Represents a product in the inventory.

  Attributes:
  `product_id`: Unique identifier for the product.
  `name`: Name of the product.
  `category`: category of the product.
  `price`: Price of the product.
  `quantity`: Quantity available in the inventory.

  Screen Shot :

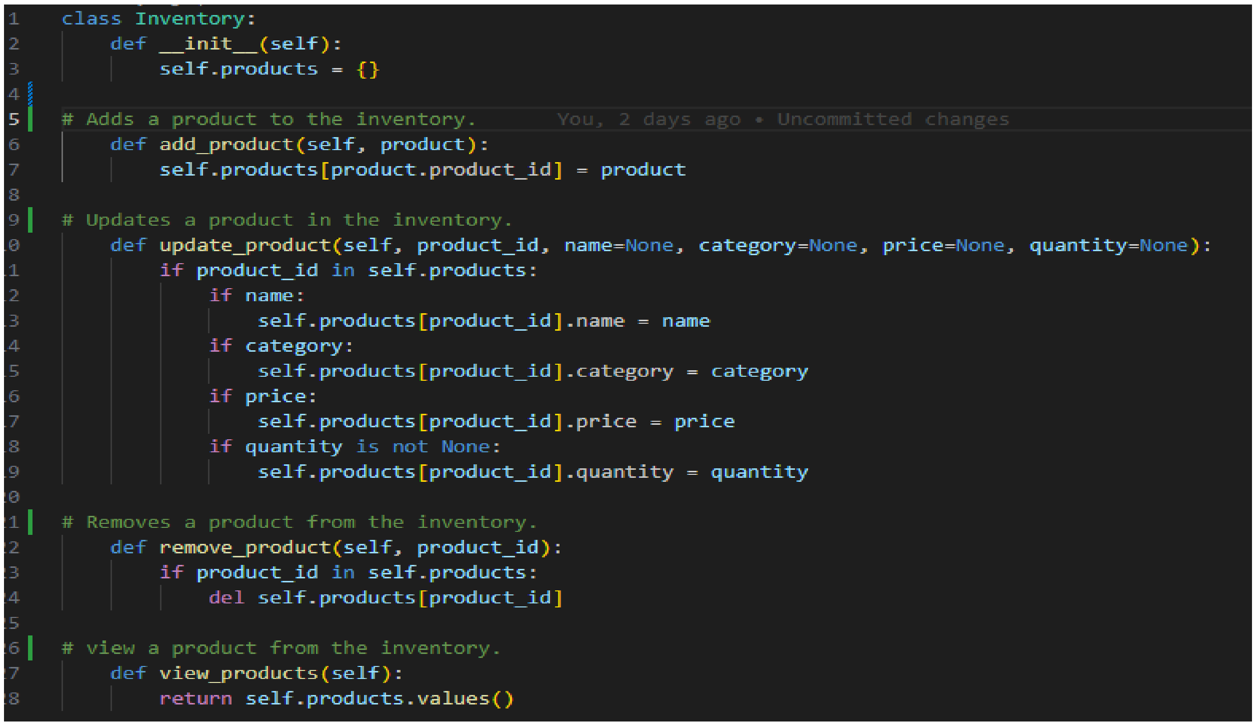


```python
class Product:        You, 1 hour ago • first commit
    def __init__(self, product_id, name, category, price, quantity):
        self.product_id = product_id
        self.name = name
        self.category = category
        self.price = price
        self.quantity = quantity
```

- Inventory Class

Attributes:
`products`: A dictionary of Product objects with `product_id` as keys.

Methods:
`add_product(self, product)`: Adds a product to the inventory.
`update_product(self,product_id, name=None, category=None, price=None, quantity=None)`: Updates the quantity of a product.
`remove_product(self,product_id)`: Removes a product from the inventory.
`view_product(self,product_id)`: view a product from the inventory.



```python
class Inventory:
    def __init__(self):
        self.products = {}

# Adds a product to the inventory.        You, 2 days ago • Uncommitted changes
    def add_product(self, product):
        self.products[product.product_id] = product

# Updates a product in the inventory.
    def update_product(self, product_id, name=None, category=None, price=None, quantity=None):
        if product_id in self.products:
            if name:
                self.products[product_id].name = name
            if category:
                self.products[product_id].category = category
            if price:
                self.products[product_id].price = price
            if quantity is not None:
                self.products[product_id].quantity = quantity

# Removes a product from the inventory.
    def remove_product(self, product_id):
        if product_id in self.products:
            del self.products[product_id]

# view a product from the inventory.
    def view_products(self):
        return self.products.values()
```

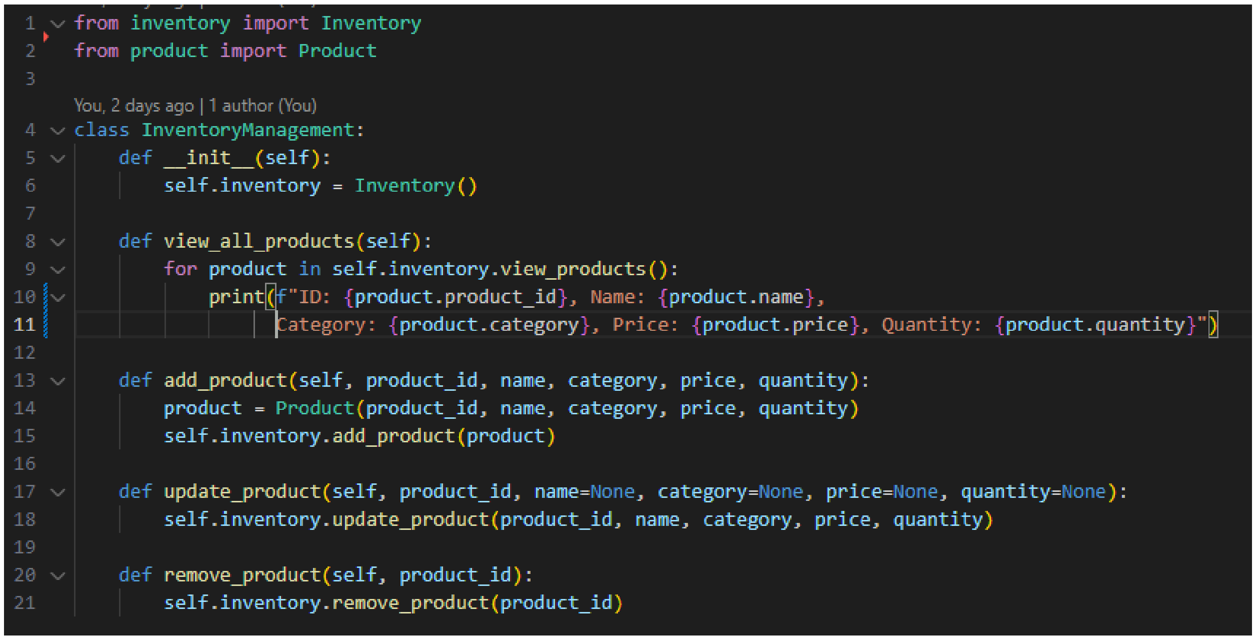
- Inventory Management Class :

Attributes:

- `inventory`: Instance of the Inventory class.

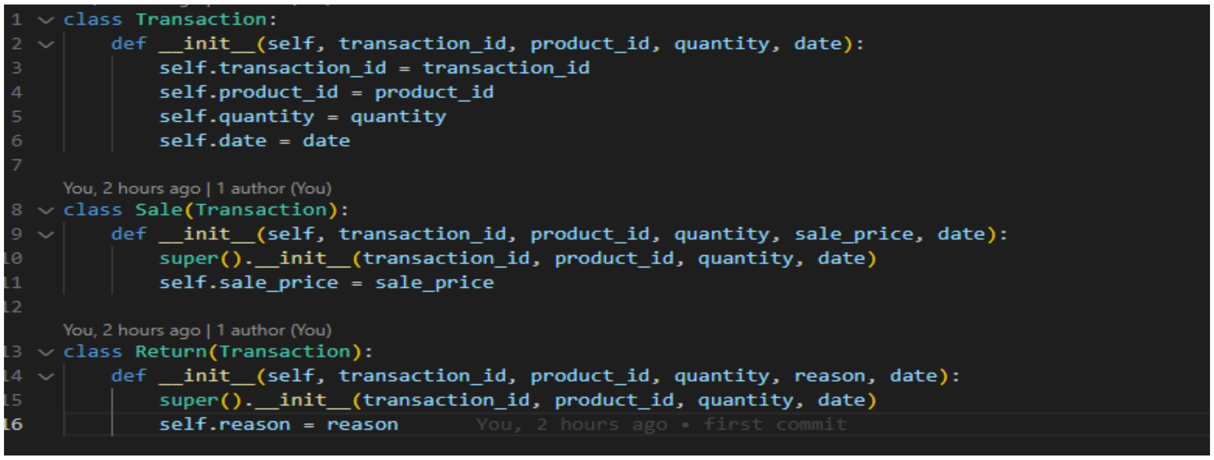
Methods:

- `add_product(self, product)`: Adds a product to the inventory.
- `update_product(self,product_id, name=None, category=None, price=None, quantity=None)`: Updates the quantity of a product.
- `remove_product(self,product_id)`: Removes a product from the inventory.
- `view_product(self,product_id)`: view a product from the inventory.



```python
from inventory import Inventory
from product import Product

You, 2 days ago | 1 author (You)
class InventoryManagement:
    def __init__(self):
        self.inventory = Inventory()

    def view_all_products(self):
        for product in self.inventory.view_products():
            print(f"ID: {product.product_id}, Name: {product.name},
                  Category: {product.category}, Price: {product.price}, Quantity: {product.quantity}")

    def add_product(self, product_id, name, category, price, quantity):
        product = Product(product_id, name, category, price, quantity)
        self.inventory.add_product(product)

    def update_product(self, product_id, name=None, category=None, price=None, quantity=None):
        self.inventory.update_product(product_id, name, category, price, quantity)

    def remove_product(self, product_id):
        self.inventory.remove_product(product_id)
```

- Transaction Class, Return Class and Sale Class

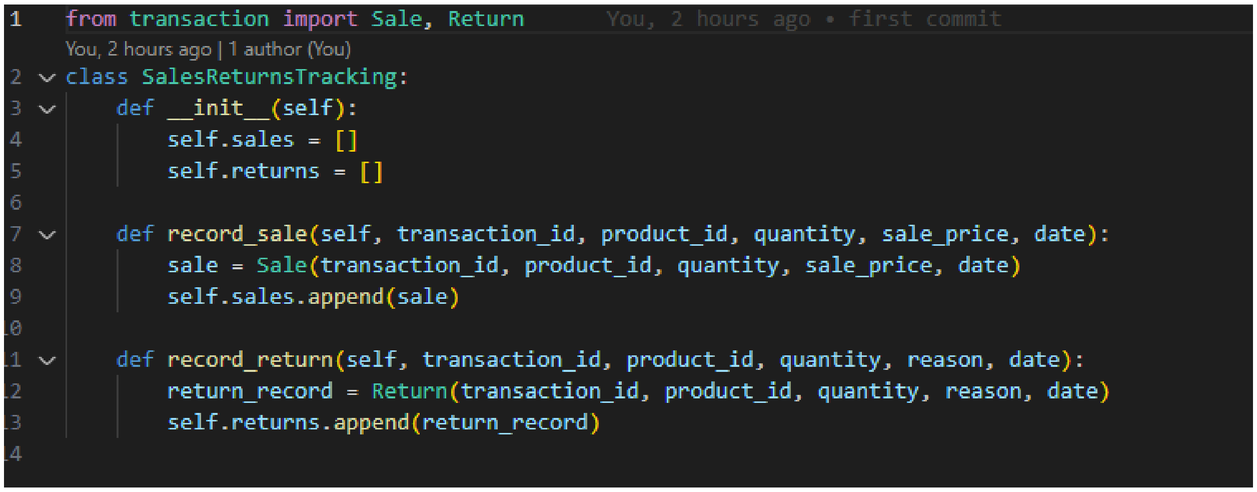


```python
class Transaction:
    def __init__(self, transaction_id, product_id, quantity, date):
        self.transaction_id = transaction_id
        self.product_id = product_id
        self.quantity = quantity
        self.date = date

You, 2 hours ago | 1 author (You)
class Sale(Transaction):
    def __init__(self, transaction_id, product_id, quantity, sale_price, date):
        super().__init__(transaction_id, product_id, quantity, date)
        self.sale_price = sale_price

You, 2 hours ago | 1 author (You)
class Return(Transaction):
    def __init__(self, transaction_id, product_id, quantity, reason, date):
        super().__init__(transaction_id, product_id, quantity, date)
        self.reason = reason        You, 2 hours ago • first commit
```

- Sales Returns tracking

  Methods:

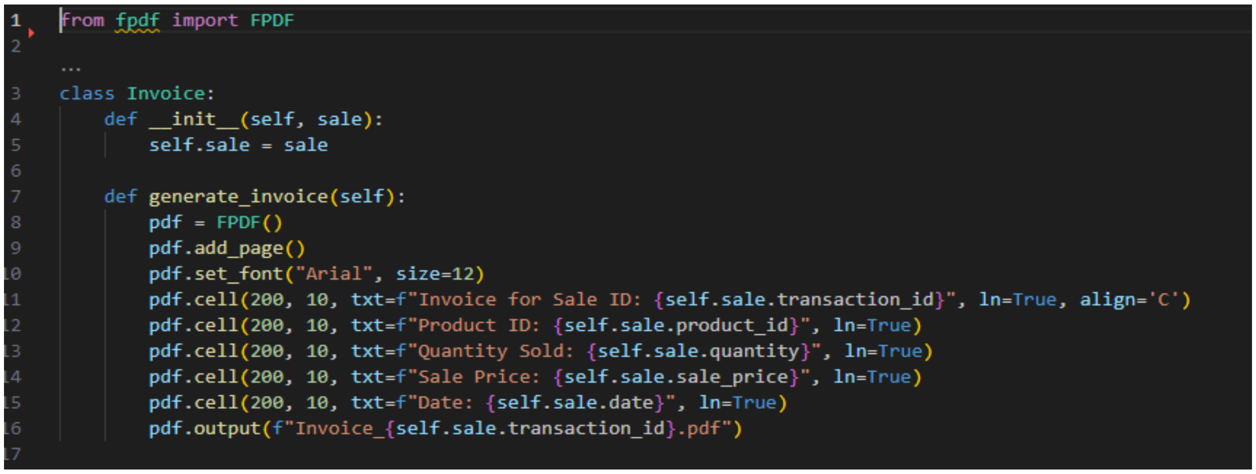
  - `record_sales(self, transaction_id, product_id, quantity, sales_price, date)`: record sales transactions.
  - `record_returns(self,, product_id, quantity, reason, date)`: record return transactions.



```
from transaction import Sale, Return        You, 2 hours ago • first commit
You, 2 hours ago | 1 author (You)
class SalesReturnsTracking:
    def __init__(self):
        self.sales = []
        self.returns = []

    def record_sale(self, transaction_id, product_id, quantity, sale_price, date):
        sale = Sale(transaction_id, product_id, quantity, sale_price, date)
        self.sales.append(sale)

    def record_return(self, transaction_id, product_id, quantity, reason, date):
        return_record = Return(transaction_id, product_id, quantity, reason, date)
        self.returns.append(return_record)

```

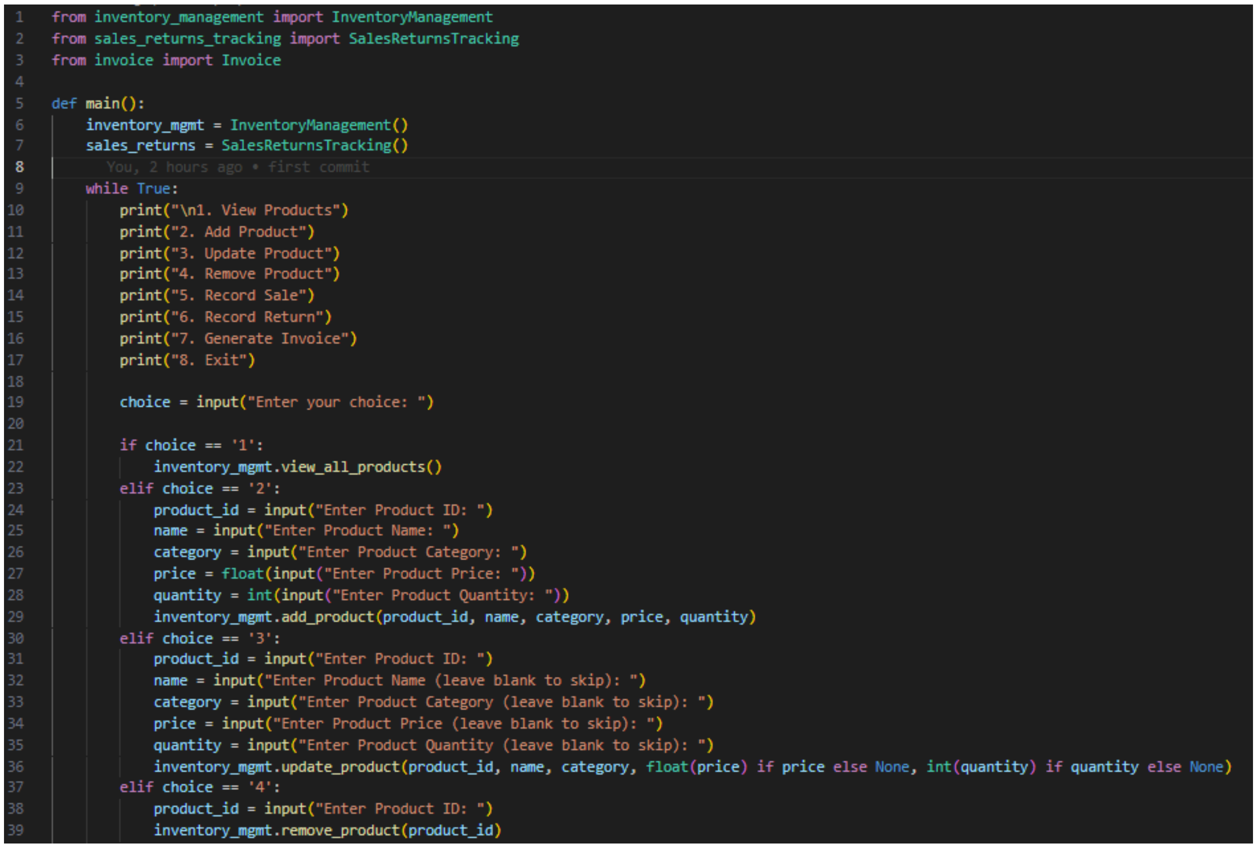
- Invoice Class

  Attributes:
  `Transaction_id,Product_id, quantity, sale_price,Total_amount,date`

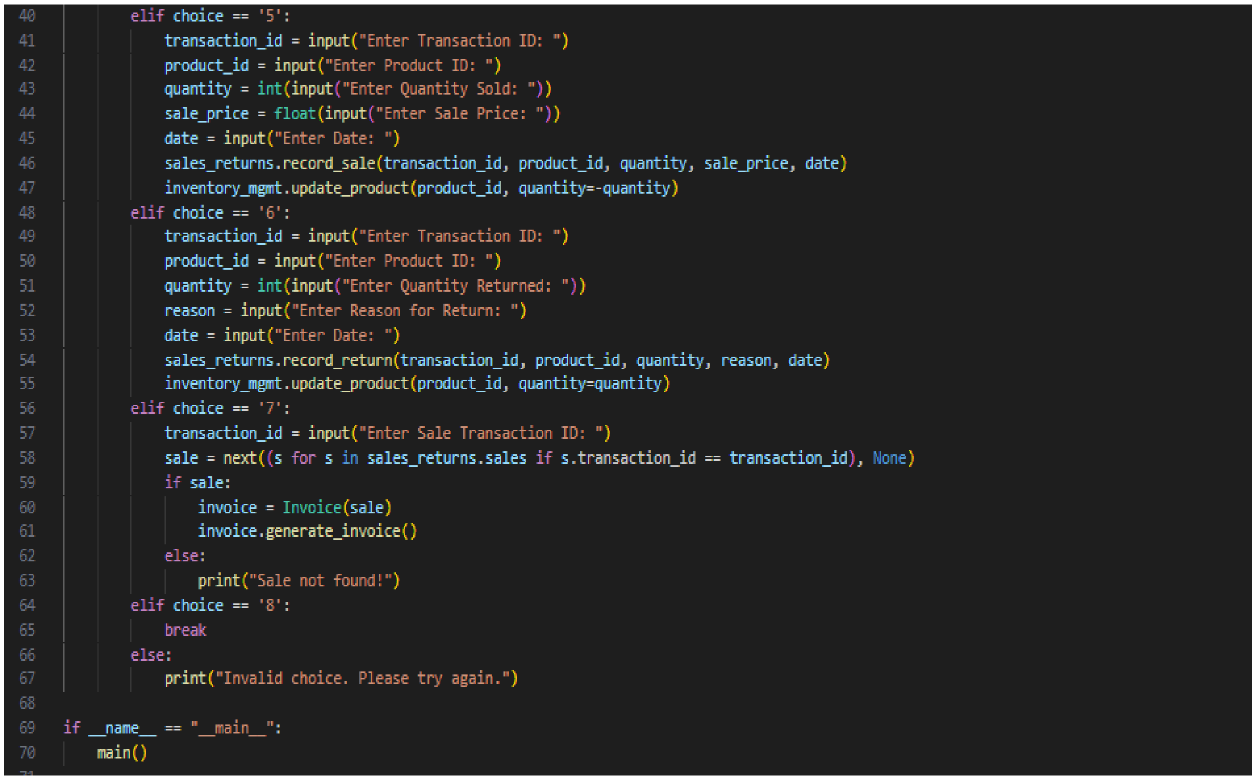


```
from fpdf import FPDF

...
class Invoice:
    def __init__(self, sale):
        self.sale = sale

    def generate_invoice(self):
        pdf = FPDF()
        pdf.add_page()
        pdf.set_font("Arial", size=12)
        pdf.cell(200, 10, txt=f"Invoice for Sale ID: {self.sale.transaction_id}", ln=True, align='C')
        pdf.cell(200, 10, txt=f"Product ID: {self.sale.product_id}", ln=True)
        pdf.cell(200, 10, txt=f"Quantity Sold: {self.sale.quantity}", ln=True)
        pdf.cell(200, 10, txt=f"Sale Price: {self.sale.sale_price}", ln=True)
        pdf.cell(200, 10, txt=f"Date: {self.sale.date}", ln=True)
        pdf.output(f"Invoice_{self.sale.transaction_id}.pdf")

```

- Main.py



```python
from inventory_management import InventoryManagement
from sales_returns_tracking import SalesReturnsTracking
from invoice import Invoice

def main():
    inventory_mgmt = InventoryManagement()
    sales_returns = SalesReturnsTracking()
        You, 2 hours ago • first commit
    while True:
        print("\n1. View Products")
        print("2. Add Product")
        print("3. Update Product")
        print("4. Remove Product")
        print("5. Record Sale")
        print("6. Record Return")
        print("7. Generate Invoice")
        print("8. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            inventory_mgmt.view_all_products()
        elif choice == '2':
            product_id = input("Enter Product ID: ")
            name = input("Enter Product Name: ")
            category = input("Enter Product Category: ")
            price = float(input("Enter Product Price: "))
            quantity = int(input("Enter Product Quantity: "))
            inventory_mgmt.add_product(product_id, name, category, price, quantity)
        elif choice == '3':
            product_id = input("Enter Product ID: ")
            name = input("Enter Product Name (leave blank to skip): ")
            category = input("Enter Product Category (leave blank to skip): ")
            price = input("Enter Product Price (leave blank to skip): ")
            quantity = input("Enter Product Quantity (leave blank to skip): ")
            inventory_mgmt.update_product(product_id, name, category, float(price) if price else None, int(quantity) if quantity else None)
        elif choice == '4':
            product_id = input("Enter Product ID: ")
            inventory_mgmt.remove_product(product_id)
```

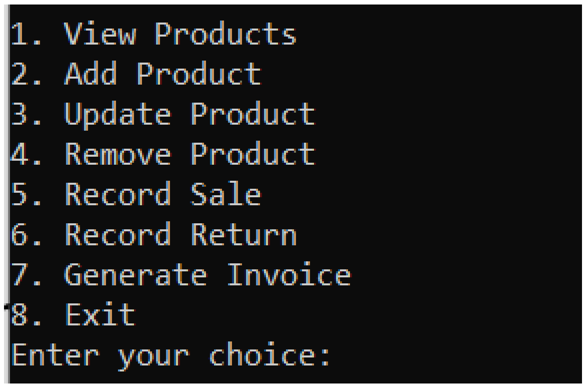


```python
        elif choice == '5':
            transaction_id = input("Enter Transaction ID: ")
            product_id = input("Enter Product ID: ")
            quantity = int(input("Enter Quantity Sold: "))
            sale_price = float(input("Enter Sale Price: "))
            date = input("Enter Date: ")
            sales_returns.record_sale(transaction_id, product_id, quantity, sale_price, date)
            inventory_mgmt.update_product(product_id, quantity=-quantity)
        elif choice == '6':
            transaction_id = input("Enter Transaction ID: ")
            product_id = input("Enter Product ID: ")
            quantity = int(input("Enter Quantity Returned: "))
            reason = input("Enter Reason for Return: ")
            date = input("Enter Date: ")
            sales_returns.record_return(transaction_id, product_id, quantity, reason, date)
            inventory_mgmt.update_product(product_id, quantity=quantity)
        elif choice == '7':
            transaction_id = input("Enter Sale Transaction ID: ")
            sale = next((s for s in sales_returns.sales if s.transaction_id == transaction_id), None)
            if sale:
                invoice = Invoice(sale)
                invoice.generate_invoice()
            else:
                print("Sale not found!")
        elif choice == '8':
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

Command Line Interface (CLI)

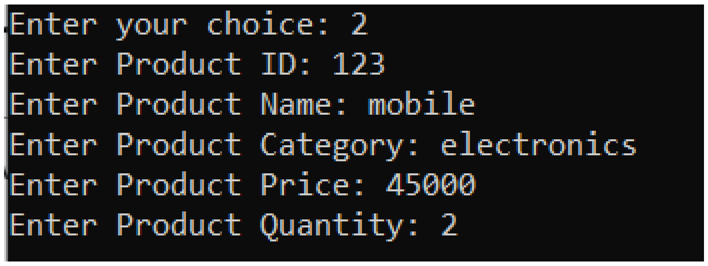
Provides a command-line interface for interaction, including commands for adding products and view, managing inventory, processing transactions, and generating invoices.

First view of Interface :

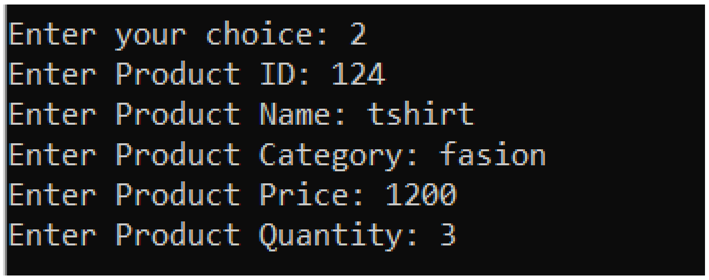


```
1. View Products
2. Add Product
3. Update Product
4. Remove Product
5. Record Sale
6. Record Return
7. Generate Invoice
8. Exit
Enter your choice:
```

Press 2 for add two products

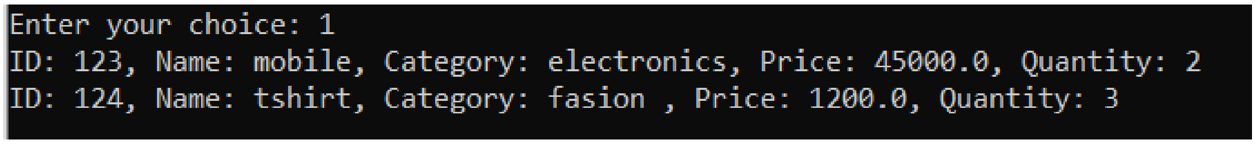


```
Enter your choice: 2
Enter Product ID: 123
Enter Product Name: mobile
Enter Product Category: electronics
Enter Product Price: 45000
Enter Product Quantity: 2
```



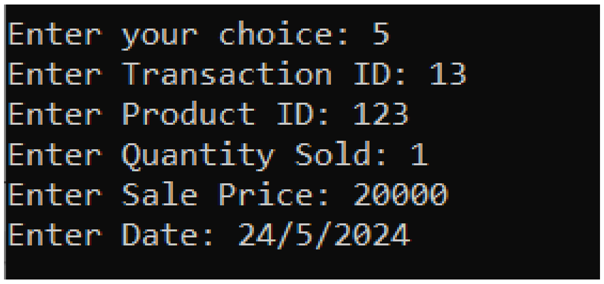
```
Enter your choice: 2
Enter Product ID: 124
Enter Product Name: tshirt
Enter Product Category: fasion
Enter Product Price: 1200
Enter Product Quantity: 3
```

Press 1 for views two added products

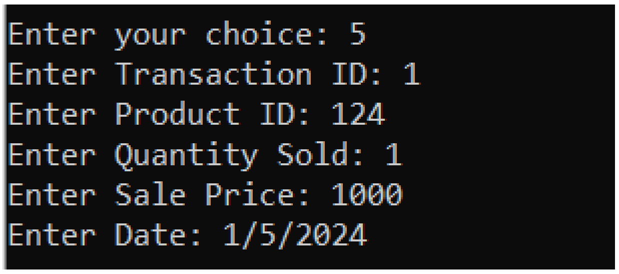


```
Enter your choice: 1
ID: 123, Name: mobile, Category: electronics, Price: 45000.0, Quantity: 2
ID: 124, Name: tshirt, Category: fasion , Price: 1200.0, Quantity: 3
```

Press 5 for record sale

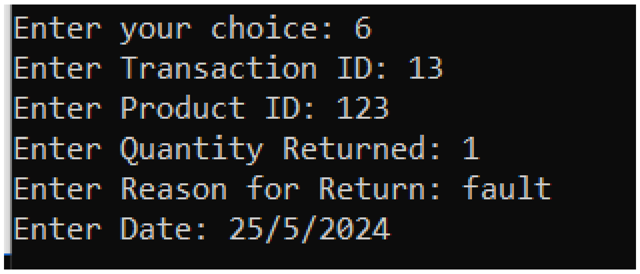


```
Enter your choice: 5
Enter Transaction ID: 13
Enter Product ID: 123
Enter Quantity Sold: 1
Enter Sale Price: 20000
Enter Date: 24/5/2024
```



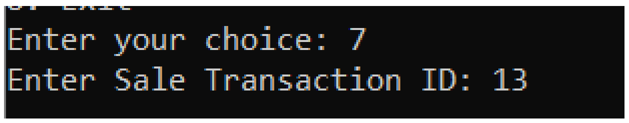
```
Enter your choice: 5
Enter Transaction ID: 1
Enter Product ID: 124
Enter Quantity Sold: 1
Enter Sale Price: 1000
Enter Date: 1/5/2024
```

Press 6 return record

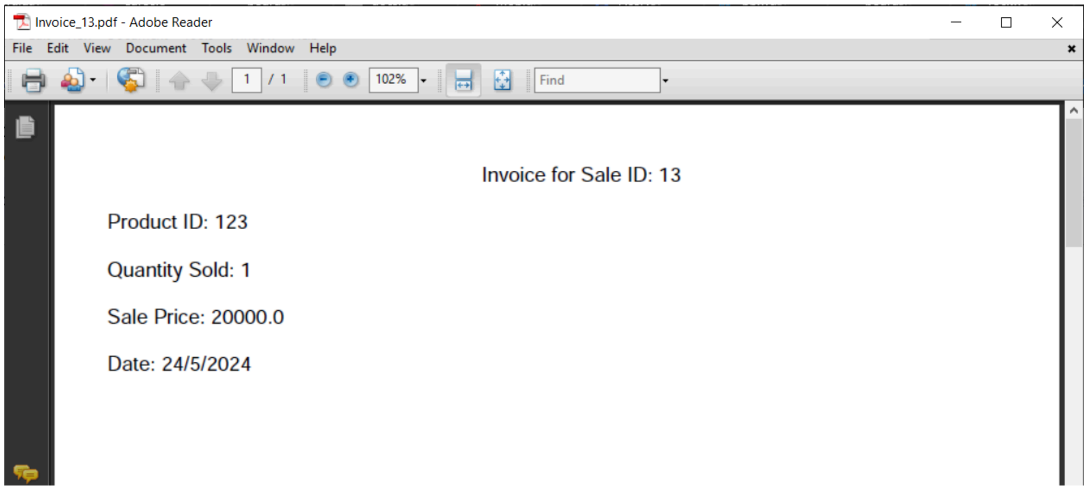


```
Enter your choice: 6
Enter Transaction ID: 13
Enter Product ID: 123
Enter Quantity Returned: 1
Enter Reason for Return: fault
Enter Date: 25/5/2024
```

Press 7 Generate PDF Invoice



```
Enter your choice: 7
Enter Sale Transaction ID: 13
```


Invoice_13.pdf - Adobe Reader
File Edit View Document Tools Window Help
1 / 1
102%
Find

Invoice for Sale ID: 13

Product ID: 123

Quantity Sold: 1

Sale Price: 20000.0

Date: 24/5/2024

Key Features and Benefits:

- **Modular Class Design:** The system is built upon well-defined classes such as `Product`, `Inventory`, `Inventory Management`, `Transaction`, `Sale`, `Return`, and `Invoice`, ensuring clear separation of concerns and facilitating easy maintenance and extensibility.
- **Efficient Inventory Management:** With functionalities for adding, updating, and removing products from inventory, along with detailed tracking of product quantities, prices, and categories, the system enables precise control over stock levels and inventory valuation.
- **Accurate Sales and Returns Tracking:** The system records sales transactions comprehensively, capturing essential details like product ID, quantity sold, sale price, and transaction date. It also manages returns efficiently, adjusting inventory quantities accordingly to maintain accuracy.
- **Professional Invoice Generation:** Leveraging the FPDF library, the system generates professional-grade PDF invoices that include itemized lists of products, quantities, prices, and total amounts. This feature enhances customer communication and financial reporting.
- **User-Friendly CLI Interface:** The command-line interface (CLI) provides a straightforward means for users to interact with the system, offering commands for performing operations such as adding products, recording sales, processing returns, and generating invoices with minimal effort.

Conclusion

In conclusion, the "Advanced Inventory Management System with Invoicing" project delivers a robust solution for modern businesses seeking streamlined inventory operations and professional invoicing capabilities. Through modular class design, efficient transaction tracking, and a user-friendly CLI interface, the system facilitates seamless management of product inventories, sales transactions, and customer invoicing.With future enhancements focused on integration, analytics, and security, this system is poised to support businesses in adapting to evolving market demands and driving sustainable growth.