

Project Report: Smart Inventory and Billing System

1. GitHub Repository

Project Source Code:

<https://github.com/AdiChakote/ds-billingsystem>

This repository contains the complete Python implementation of the **Smart Inventory and Billing System**, including inventory management, billing logic, and role-based access for customers and shopkeepers.

2. Introduction

The **Smart Inventory and Billing System** is a console-based Python application designed to simulate a basic **retail store environment**.

The system supports two roles:

- **Customer** – for purchasing items and generating bills
- **Shopkeeper** – for managing stock and adding new items

This project demonstrates real-world concepts such as **inventory management, billing logic, discounts, and role-based access** using Python.

3. Objective

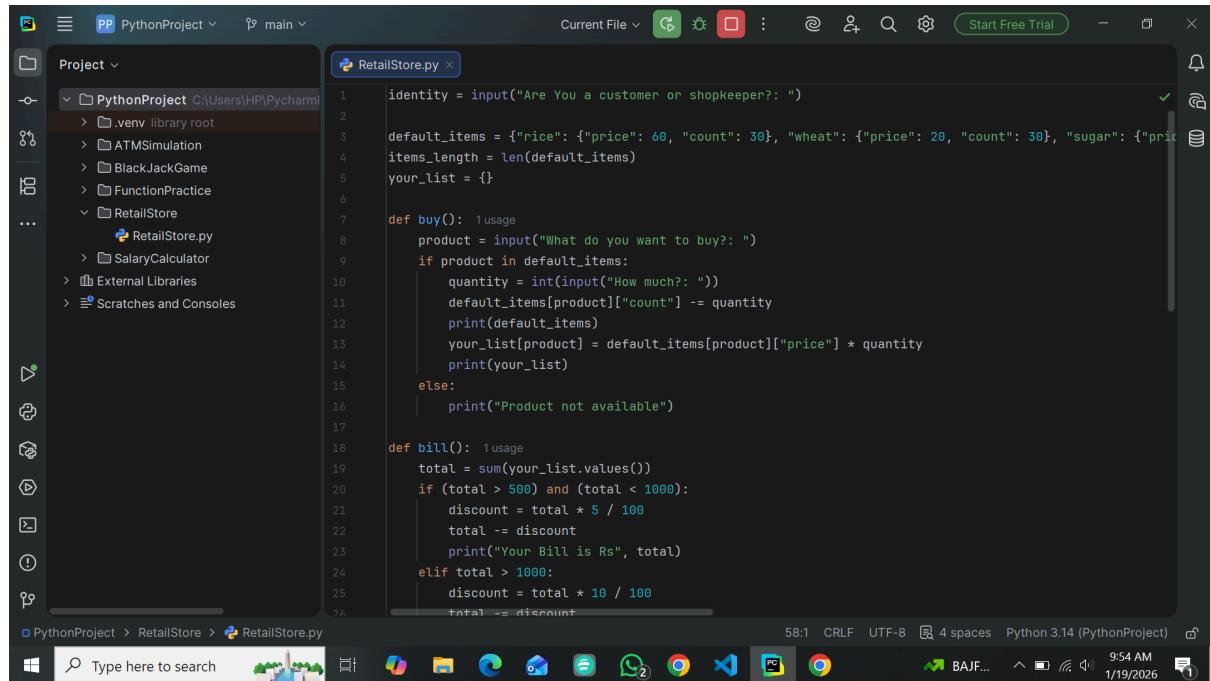
The main objectives of this project are:

- To manage store inventory using Python dictionaries
 - To allow customers to purchase items and generate bills
 - To provide discount logic based on total purchase amount
 - To allow shopkeepers to view and update stock
 - To demonstrate modular programming using functions
-

4. Technologies Used

- **Programming Language:** Python
 - **Concepts Applied:**
 - Dictionaries (nested dictionaries)
 - Functions
 - Conditional statements
 - Loops
 - User input/output
 - **Interface:** Command Line (CLI)
-

5. Source Code



The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure for 'PythonProject' containing files like 'RetailStore.py', 'SalaryCalculator.py', and 'External Libraries'. The main editor window shows the code for 'RetailStore.py'. The code defines a function 'buy()' which asks for user input to identify as a customer or shopkeeper, then prompts for a product name and quantity. It checks if the product is available in a dictionary of default items. If available, it calculates the total price and updates the item's count. If not available, it prints a message. The code then defines a function 'bill()' which calculates the total bill based on the items purchased, applying discounts for purchases over 500 and 1000.

```
identity = input("Are You a customer or shopkeeper?: ")

default_items = {"rice": {"price": 60, "count": 30}, "wheat": {"price": 20, "count": 30}, "sugar": {"price": 10, "count": 30}}
items_length = len(default_items)
your_list = {}

def buy():
    usage
    product = input("What do you want to buy?: ")
    if product in default_items:
        quantity = int(input("How much?: "))
        default_items[product]["count"] -= quantity
        print(default_items)
        your_list[product] = default_items[product]["price"] * quantity
        print(your_list)
    else:
        print("Product not available")

def bill():
    usage
    total = sum(your_list.values())
    if (total > 500) and (total < 1000):
        discount = total * 5 / 100
        total -= discount
        print("Your Bill is Rs", total)
    elif total > 1000:
        discount = total * 10 / 100
        total -= discount
        print("Your Bill is Rs", total)
```

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** PythonProject C:\Users\HP\Pycharm
- File:** RetailStore.py
- Code Content:**

```
18     def bill(): 1usage
19         total -= discount
20         print("Your Bill is Rs", total)
21     else:
22         print("Your Bill is Rs", total)
23
24 def stock_management(): 1usage
25     item = input("Name of Item: ")
26     price = int(input("Cost of Item?: "))
27     count = int(input("Quantity of Item?: "))
28     default_items[item] = {"price": price, "count": count}
29     print(default_items)
30
31 if identity == "customer":
32     print(default_items)
33     choice = input("Enter 'p' for Purchase or 'b' for Bill: ")
34     for i in range(items_length):
35         if choice == "p":
36             buy()
37         choice = input("Enter 'p' for Purchase or 'b' for Bill: ")
38         if choice == "b":
39             bill()
40     elif identity == "shopkeeper":
41         work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
42         for i in range(items_length):
43             if work == "s":
44                 print(default_items)
45             work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
46             elif work == "i":
47                 stock_management()
48             work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
```

- Toolbars and Status Bar:** Current File, Start Free Trial, 58:1 CRLF, UTF-8, 4 spaces, Python 3.14 (PythonProject), 9:54 AM, 1/19/2026.

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** PythonProject C:\Users\HP\Pycharm
- File:** RetailStore.py
- Code Content:**

```
39     print(default_items)
40     choice = input("Enter 'p' for Purchase or 'b' for Bill: ")
41     for i in range(items_length):
42         if choice == "p":
43             buy()
44             choice = input("Enter 'p' for Purchase or 'b' for Bill: ")
45             if choice == "b":
46                 bill()
47     elif identity == "shopkeeper":
48         work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
49         for i in range(items_length):
50             if work == "s":
51                 print(default_items)
52                 work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
53             elif work == "i":
54                 stock_management()
55             work = input("Do you want to see Stock 's' or Add Items 'i' in store?: ")
```

- Toolbars and Status Bar:** Current File, Start Free Trial, 56:1 CRLF, UTF-8, 4 spaces, Python 3.14 (PythonProject), 9:55 AM, 1/19/2026.

📸 Screenshot: Smart Inventory and Billing System Code

6. Code Overview

```
identity = input("Are You a customer or shopkeeper?: ")
```

```
default_items = {  
    "rice": {"price": 60, "count": 30},  
    "wheat": {"price": 20, "count": 30},  
    "sugar": {"price": 40, "count": 70},  
    "ghee": {"price": 600, "count": 40},  
    "soap": {"price": 30, "count": 100},  
    "toothpaste": {"price": 50, "count": 80}  
}  
  
your_list = {}
```

7. Data Structure Explanation

7.1 Inventory Dictionary (`default_items`)

- Stores product name as key
- Each product has:
 - `price`
 - `count` (available stock)

Example:

```
"rice": {"price": 60, "count": 30}
```

7.2 Cart Dictionary (`your_list`)

- Stores purchased items and their total cost
 - Used for bill calculation
-

8. Function-wise Explanation

8.1 buy() Function

```
def buy():  
  
    product = input("What do you want to buy?: ")  
  
    if product in default_items:  
  
        quantity = int(input("How much?: "))  
  
        default_items[product]["count"] -= quantity  
  
        your_list[product] = default_items[product]["price"] *  
        quantity  
  
    else:  
  
        print("Product not available")
```

Purpose:

- Allows customer to purchase products

Logic:

- Checks if product exists
 - Reduces stock count
 - Adds item to customer cart
-

8.2 bill() Function

```

def bill():

    total = sum(your_list.values())

    if (total > 500) and (total < 1000):

        discount = total * 5 / 100

        total -= discount

    elif total > 1000:

        discount = total * 10 / 100

        total -= discount

    print("Your Bill is Rs", total)

```

Purpose:

- Calculates final bill amount

Discount Rules:

- ₹500 – ₹999 → **5% discount**
 - Above ₹1000 → **10% discount**
 - Below ₹500 → No discount
-

8.3 stock_management() Function

```

def stock_management():

    item = input("Name of Item: ")

    price = int(input("Cost of Item?: "))

    count = int(input("Quantity of Item?: "))

    default_items[item] = {"price": price, "count": count}

```

Purpose:

- Allows shopkeeper to add or update items in inventory
-

9. Role-Based Logic

9.1 Customer Flow

```
if identity == "customer":
```

Customer can:

- View available items
 - Purchase products
 - Generate bill
-

9.2 Shopkeeper Flow

```
elif identity == "shopkeeper":
```

Shopkeeper can:

- View stock
 - Add new items
 - Update inventory
-

10. Sample Execution

Example:

Are You a customer or shopkeeper?: customer

What do you want to buy?: rice

How much?: 5

Your Bill is Rs 300

11. Features of the System

- Role-based access (Customer / Shopkeeper)
 - Real-time stock update
 - Discount-based billing system
 - Modular and reusable functions
 - Dictionary-based inventory storage
-

12. Limitations

- No validation for negative or excess quantity
 - No loop exit option
 - Stock can go negative
 - No persistent storage (data resets on restart)
 - No authentication for shopkeeper
-

13. Future Scope / Enhancements

The system can be enhanced by:

1. Input Validation

- Prevent negative values
- Check stock availability before purchase

2. Persistent Storage

- Save data using files or databases

3. Multiple Customers

- Support multiple users and sessions

4. Authentication System

- Login system for shopkeeper

5. Advanced Billing

- GST, tax, invoice generation

6. GUI / Web Application

- Convert to Tkinter / Flask / Django app
-

14. Conclusion

The Smart Inventory and Billing System effectively demonstrates real-world retail logic using Python. It applies core programming concepts such as dictionaries, functions, and conditional logic in a practical scenario. This project serves as a strong foundation for building more advanced inventory and billing systems.