**Enhanced E-commerce Shopping Cart Project Report**

1. **Project Overview**

The main purpose of this project was to design and implement a shopping cart system for an e-commerce platform. This system was built using Object-Oriented Programming (OOP) principles by using key concepts such as Encapsulation, Inheritance, Abstraction, and Polymorphism.

The project consists of four main classes:

Product class : Handling different types of products (digital and physical) using the Product class and its derived classes.

Cart System : Managing user carts, adding/removing items, and calculating totals through the Cart class.

User class: Manages user detail and cart interaction

Discount System : Applying discounts to purchases using the Discount class and its subclasses.

1. **Project Overview**

| Product |
| --- |
| Attributes   * product\_id * name * Price * quantity   Methods   * update\_quantity(self,new\_quantity) update the quantity to new value * get\_product\_info(self) returns product details (id,name,price,quantity) |

^

|

|

| Digital Product (derived from Product ) | Physical Product (derived from Product) |
| --- | --- |
| Attributes  (get the attributes from product using super())   * file\_size * download\_link   Methods   * get\_product\_info()   Overrides method from Product class  To include file size and download  link . | Attributes  (get the attributes from product using super())   * weight * dimension * shipping\_cost   Methods   * get\_product\_info()   Overrides method from Product class  To include file size and download  link . |

| Cart |
| --- |
| Attributes   * \_\_cart\_items   Methods   * add\_product(self,product) Adds a product to the cart * remove\_product(self,product\_id) removes a product from the cart * view\_cart(self) display all product in the cart * calculate\_total() calculate the total cost of all product in cart |

| User |
| --- |
| Attributes   * user\_id * name * cart (Cart instance)   Methods   * add\_to\_cart(self,product) adds a product to the user’s cart * remove\_from\_cart(self,product\_id) removes a product from the user’s cart * checkout(self,discount=None) calculate total, applies a discount and clear the cart |

| Discount (Parent Abstract class) |
| --- |
| Method   * apply\_disocunt(self,total\_amount) abstract method to apply a discount |

| PercentageDiscount (child class derived from discount ) |
| --- |
| Attributes   * percentage   Method   * apply\_discount(self, total\_amount) applies percentage discount |

| FixedAmountDiscount (child class derived from discount ) |
| --- |
| Attributes   * amount   Method   * apply\_discount(self, total\_amount) applies fixed amount discount |

**Brief summary on developed classes**

1. Product (Parent class) : The products class is a foundation for all product types. It Stores product details. It has a method to update the quantity and another method to print product details.
2. DigitalProduct (child class Inherits from Product ): The DigitalProduct class extends Product by adding new attributes specific to digital products like file\_size and download\_link. It overrides the method to display the extra attributes of this class.
3. PhysicalProduct (child class Inherits from Product ): The PhysicalProduct class extends Product by adding attributes like weight, dimension and shipping cost. It overrides the method to display the extra attributes of this class.
4. Cart: The Cart class stores a collection of products for a user. It has multiple methods like adding/ removing items ,calculating the total cost and a method to view the user’s cart. It keeps the cart items private.
5. User: The user class uses abstraction to hide the complexity of cart operations from the user. Each user has an id, name and their own cart. It has methods like add/remove product from cart and a checkout method which prints the total after discount applied.
6. Discount (Abstract base class): The discount class defines a common structure for other discount classes. It forces other discount classes to have the apply discount method.
7. PercentageDiscount(Inherits from Discount): Applies a percentage discount and uses an abstract method from the discount class.
8. FixedAmountDiscount(Inherits from Discount): Applies a fixed amount discount and uses an abstract method from the discount class.
9. **Instruction for Use**

1. Create Products Instances (Digital or Physical product)

2. Create User instances and give a cart to each user

3. Add product to the cart using add\_to\_cart() method

4.View the user cart using view\_cart() method

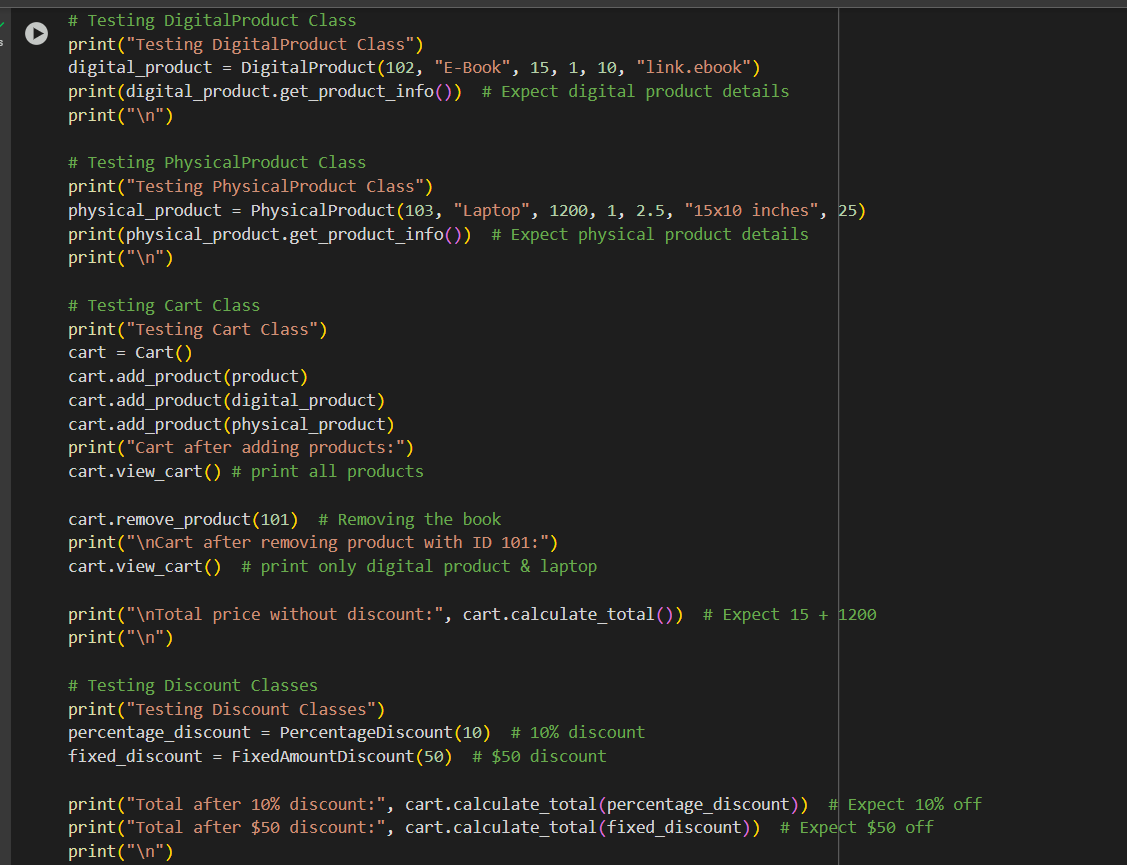
5.Apply discount if the user have any

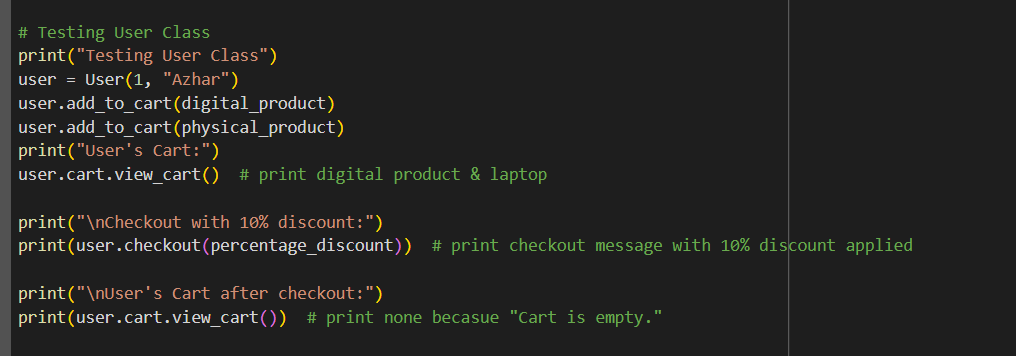
6.Then do the checkout, which gives the total to the user after discount is applied.

1. **Verification Of Code Functionality**

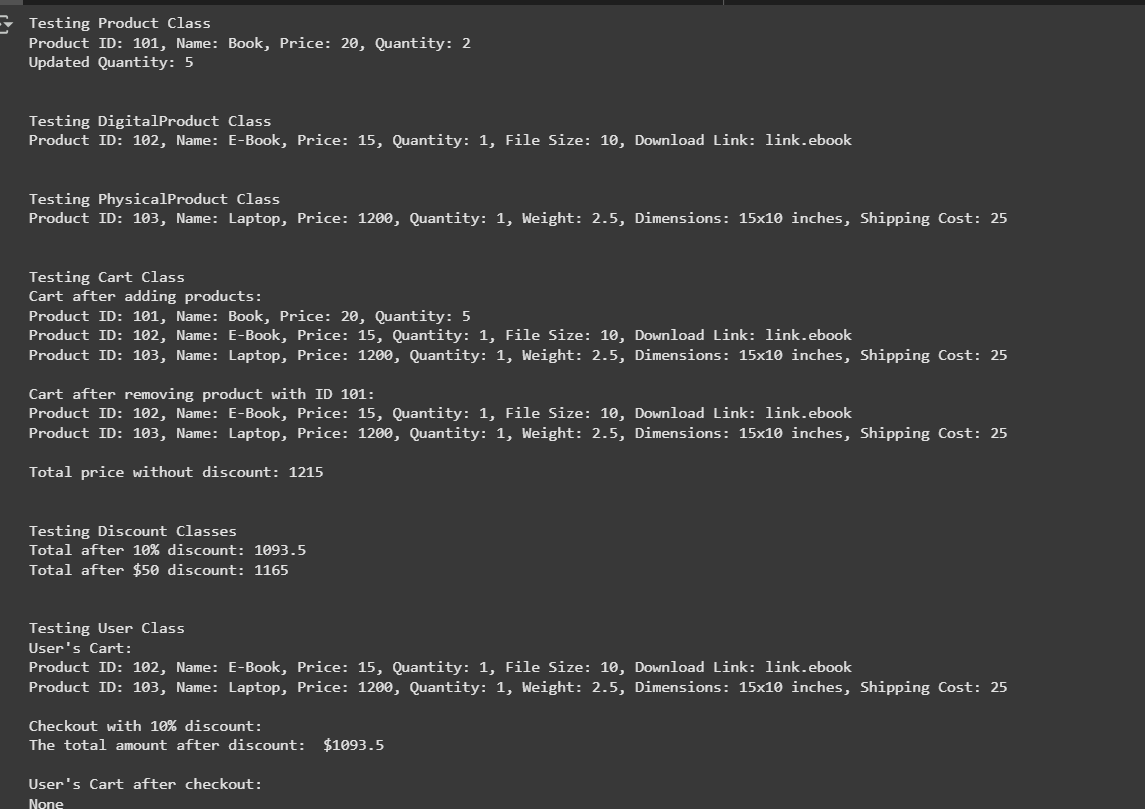
* Include screenshots or examples of code execution (Here I took picture of using every method)

Code

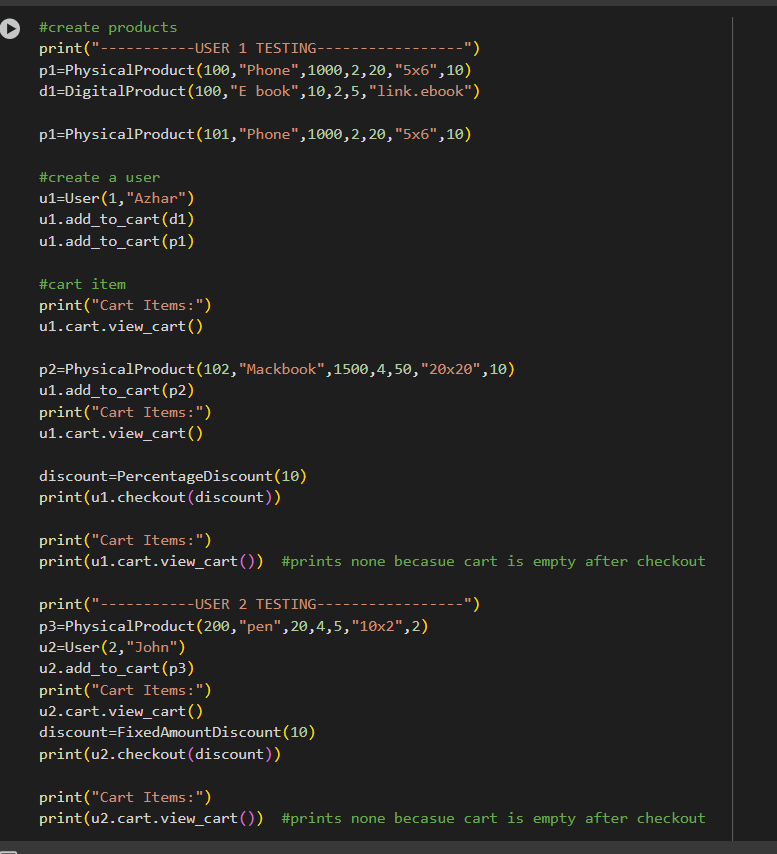




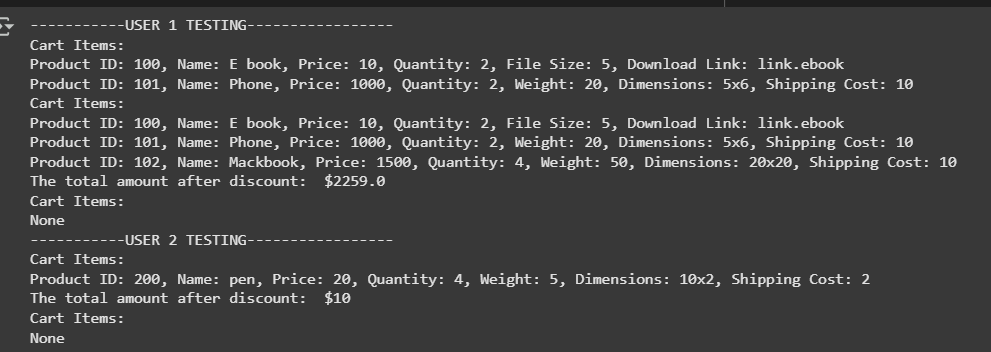
Result



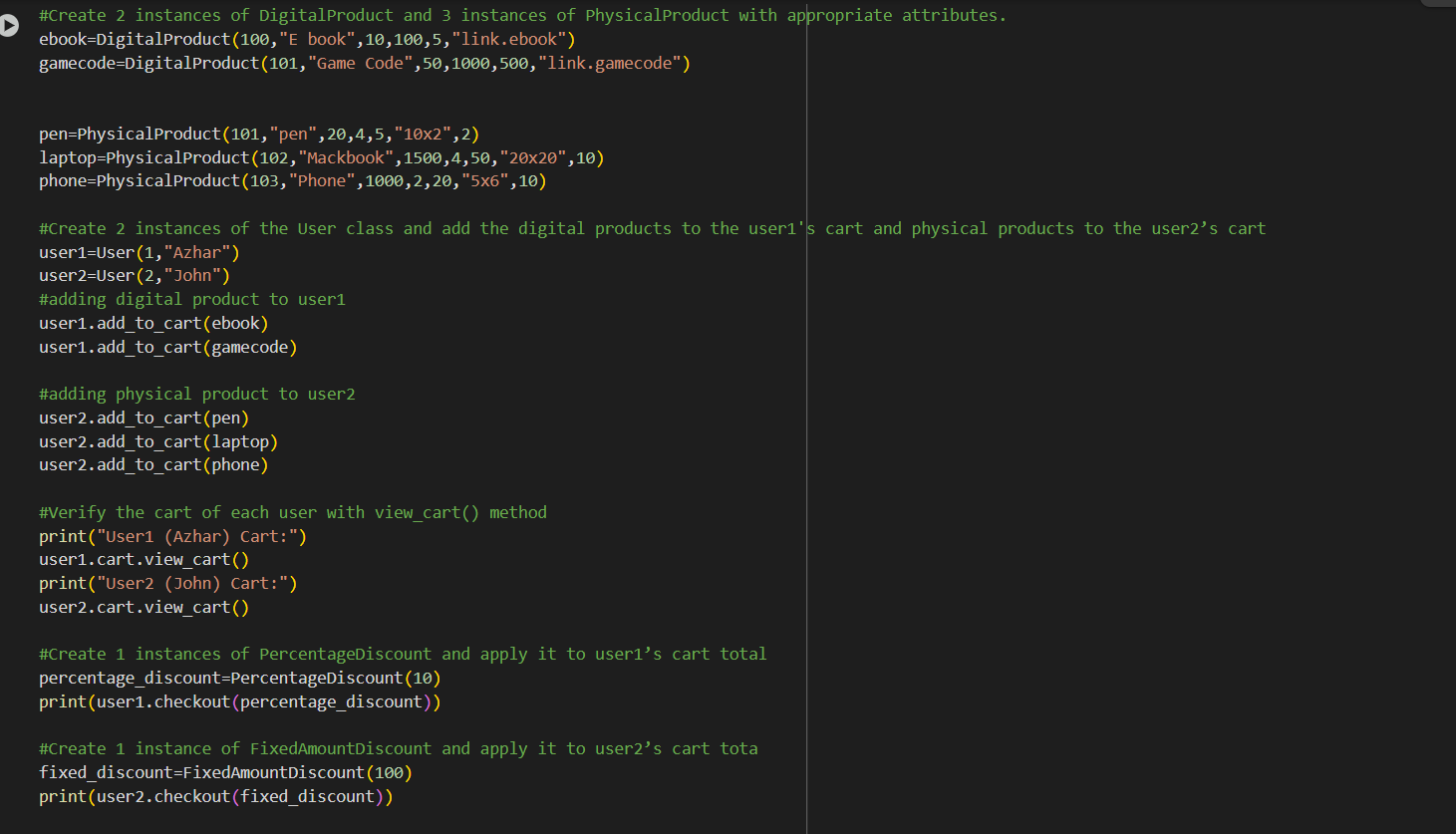
Code execution

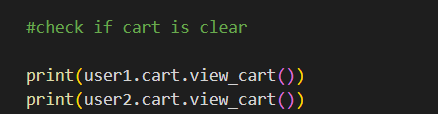


Result

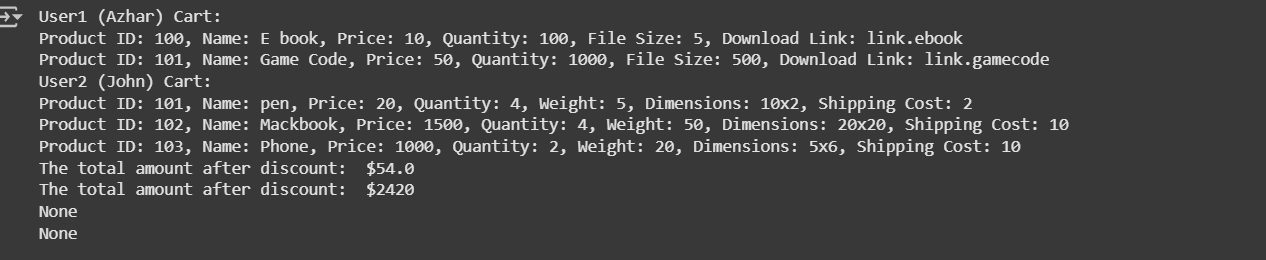


**Required Sample Scenario**

****

****

**Result**

****

**5.Discussion and Conclusion**

Your finding through this project

1. This project showed how effective OOP is in these types of projects . OOP helps users for code reusability, easier maintenance and makes everything organized. This project made me realize why almost every big company utilizes OOP for their big projects. Through this project, I realized the importance of encapsulation, especially in keeping the cart private. This shows how we have to use a public method to access the private attribute of the user's cart. This makes sure the user's data is secured. Implementing inheritance and polymorphism made the system more efficient and it also saves lots of time as I could reuse the code of the parent class. Also, using Abstraction in Discount class showed me how it serves as a blueprint for other classes and it forces those classes to implement the method of the parent abstract class. Overall this project improved my understanding of OOP concepts and their real world applications.

Challenges Faced

1. While working on this project I faced many challenges like learning abstraction through YouTube tutorials and then applying that knowledge to the project. Another challenge was planning before coding, which initially made my code unorganized and wasted time. As a result, I had to put extra effort to structure the code properly.

Area for improvement

1. One area of improvement would be if a user tries to add more products than the available stock, the system should give a better message. We can maybe use “try/Catch” for this. Another improvement is that After a user buys the product , the product quantity should go down by itself. Finally, another improvement is to add user login functionality which would allow the system to store user data.