Advanced Programming

Season 2024-III

Report of Workshop No. 1

Santiago Andrés Benavides Coral

20232020036

Systems engineer

Universidad Distrital Francisco José de Caldas

User stories:

1. CONSULTATION OF PRODUCTS AVAILABLE FOR PURCHASE.
As any student I can consult the list of products available to purchase.

2. CONSULTATION OF PURCHASE CATEGORIES.
As any student I can consult the category of products available to purchase.

3. CONSULTATION OF PRODUCT FOR SYSTEMS ENGINEER STUDENTS.
As systems engineer I can consult the products corresponding to the systems engineering category to know what products I can buy.

4. CONSULTATION OF PRODUCT FOR ELECTRONIC ENGINEER STUDENTS.
As electronic engineer I can consult the products corresponding to the electronic engineering category to know what products I can buy.

5. CONSULTATION OF PRODUCT FOR ARCHITECTURE STUDENTS.
As architect I can consult the products corresponding to the architect category to know what products I can buy.

6. PURCHASE OF PRODUCTS
As a buyer I can select the products I want to buy and add them to my shopping cart to proceed with checkout.

7. PURCHASE CHECKOUT
As a buyer I can check out my shopping cart and proceed to fill out the information to proceed with the payment and delivery of the products.

Object-oriented principles analysis:

This program showcases the application of Object-Oriented Programming (OOP) principles in Python. Below is an analysis of the key OOP principles as applied to this program:

1. ENCAPSULATION.

Encapsulation involves keeping the fields within a class private and providing access through public methods. This helps bundle data with the methods that operate on it and control access.

Product: Encapsulates product details (product_id, name, category, price) and includes a method to represent the product as a string.

ProductRepository: Manages the list of products and hides the implementation details of loading products from a CSV file. It provides methods to interact with the product list.

ShoppingCart: Maintains a list of cart items and offers methods to add products, calculate totals, and list items.

Checkout: Manages the checkout process and interacts with the ShoppingCart object, encapsulating the checkout logic and user input handling.

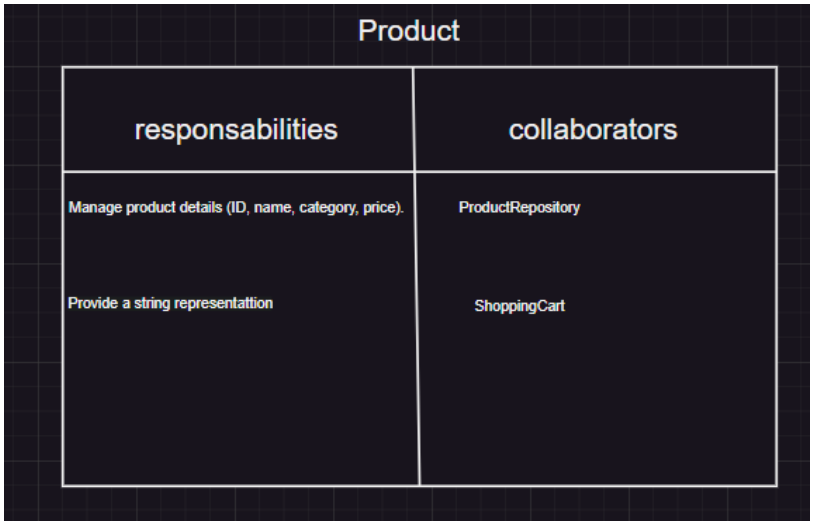
2. ABSTRACTION.

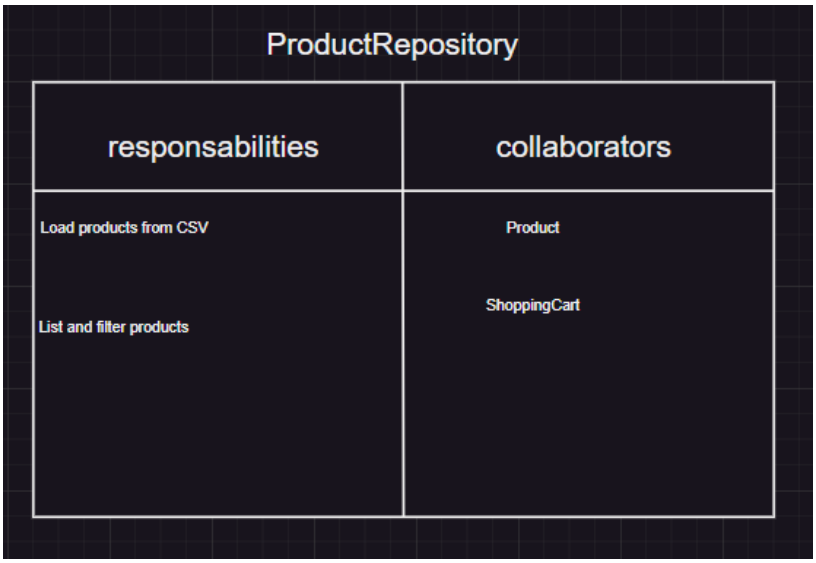
Abstraction hides complex implementation details and shows only the necessary features of an object, helping manage complexity by breaking down the system into more manageable parts.

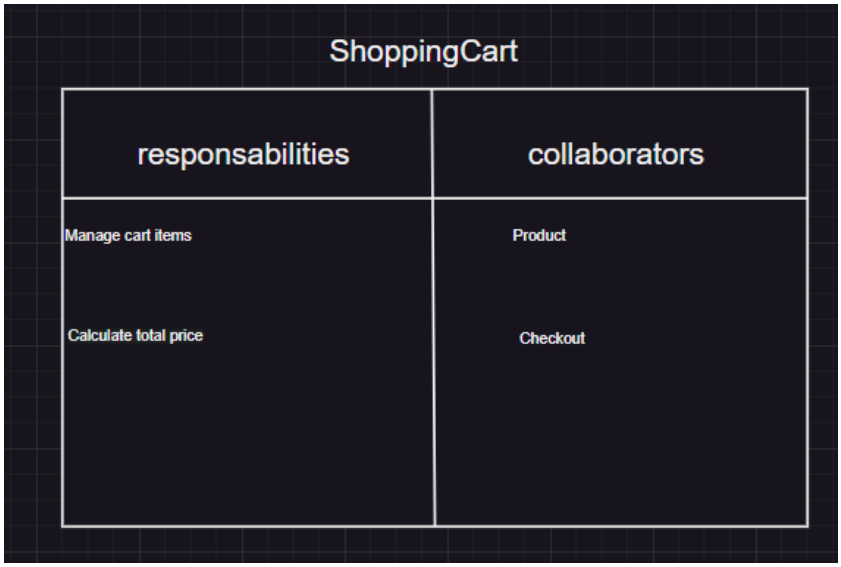
ProductRepository: Abstracts the details of loading products from a CSV file. Users interact with the repository through methods like list_all_products and list_products_by_category without needing to know how the data is loaded or parsed.

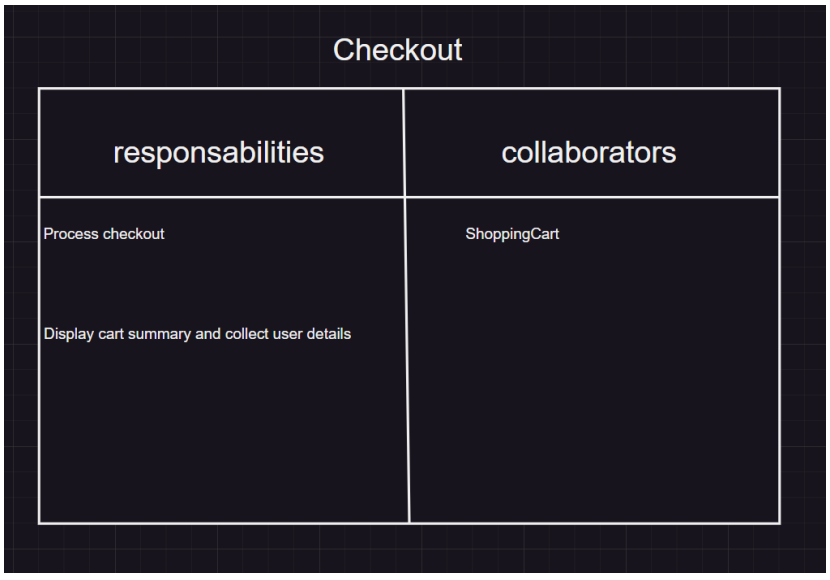
ShoppingCart: Abstracts the complexity of managing the cart and calculating totals. Users interact with a simplified interface provided by methods like add_product, calculate_total, and list_cart_items.

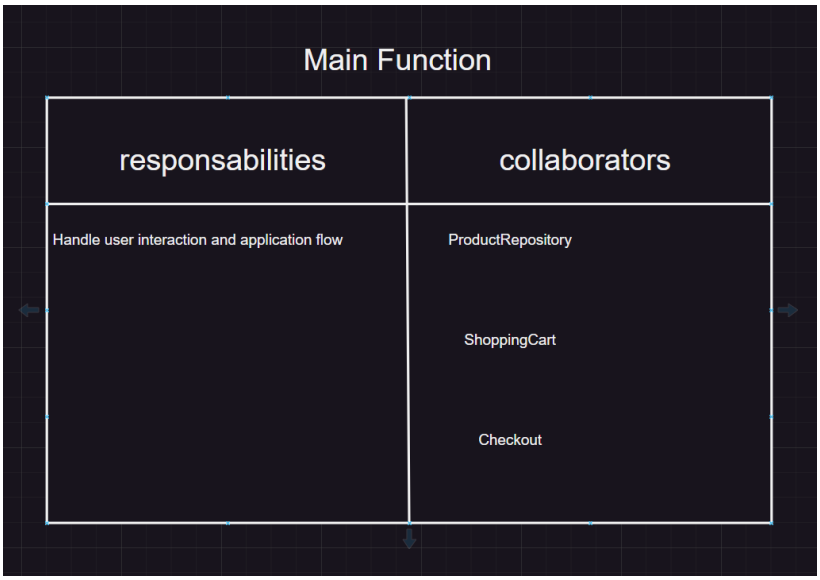
CRC cards:



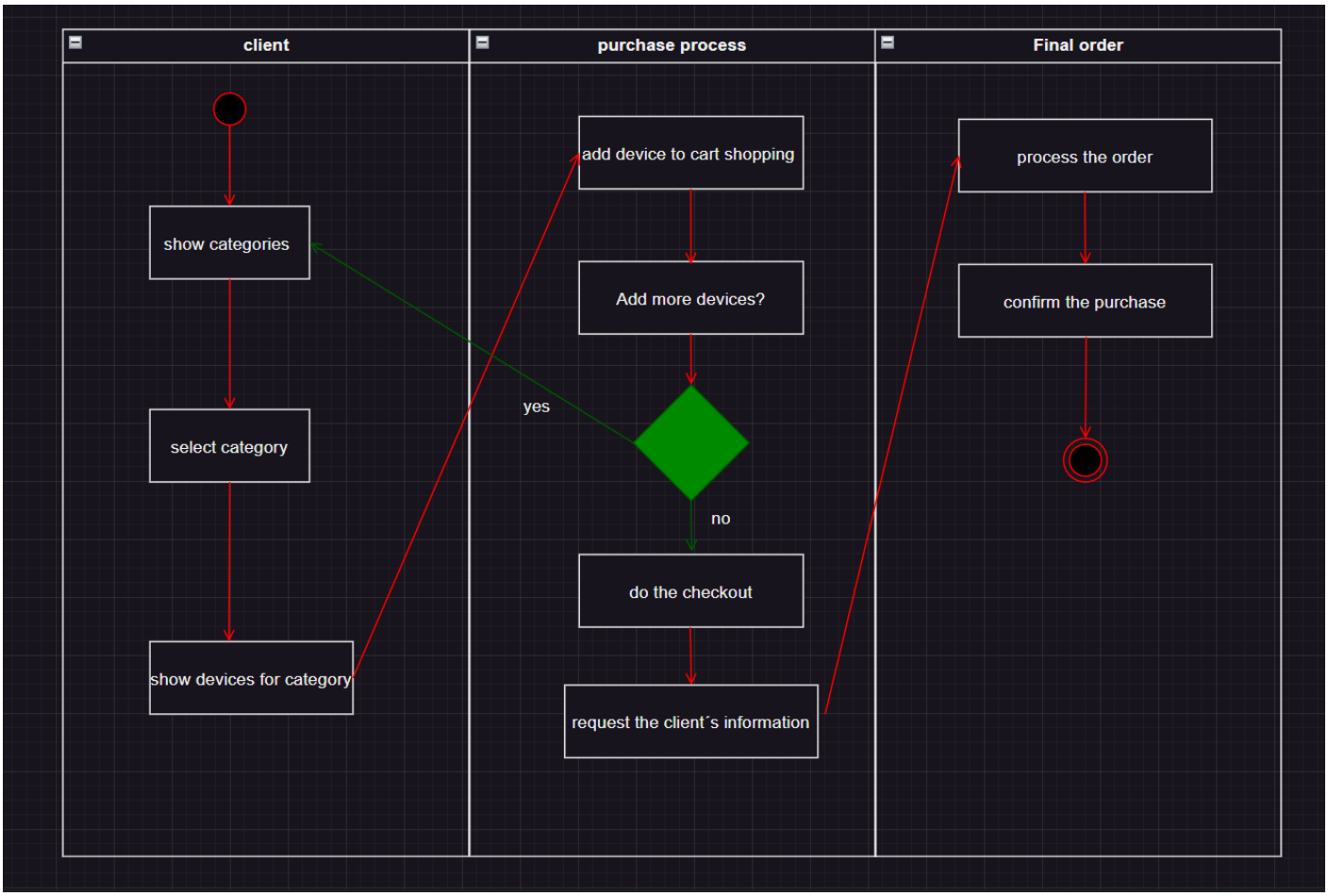

ProductRepository
responsabilities
collaborators
Load products from CSV
List and filter products
Product
ShoppingCart


ShoppingCart
responsabilities
collaborators
Manage cart items
Calculate total price
Product
Checkout

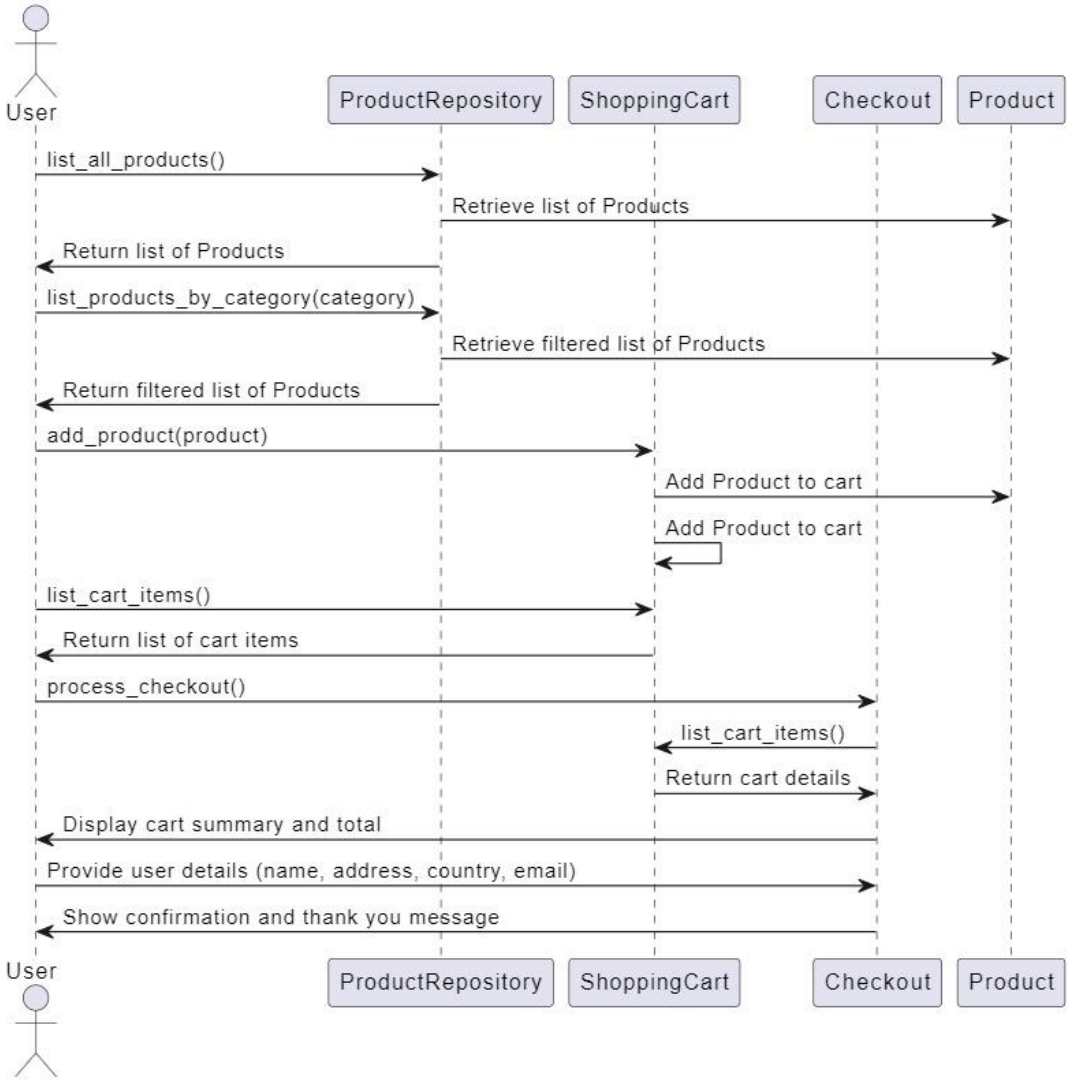

Checkout
responsabilities
collaborators
Process checkout
Display cart summary and collect user details
ShoppingCart



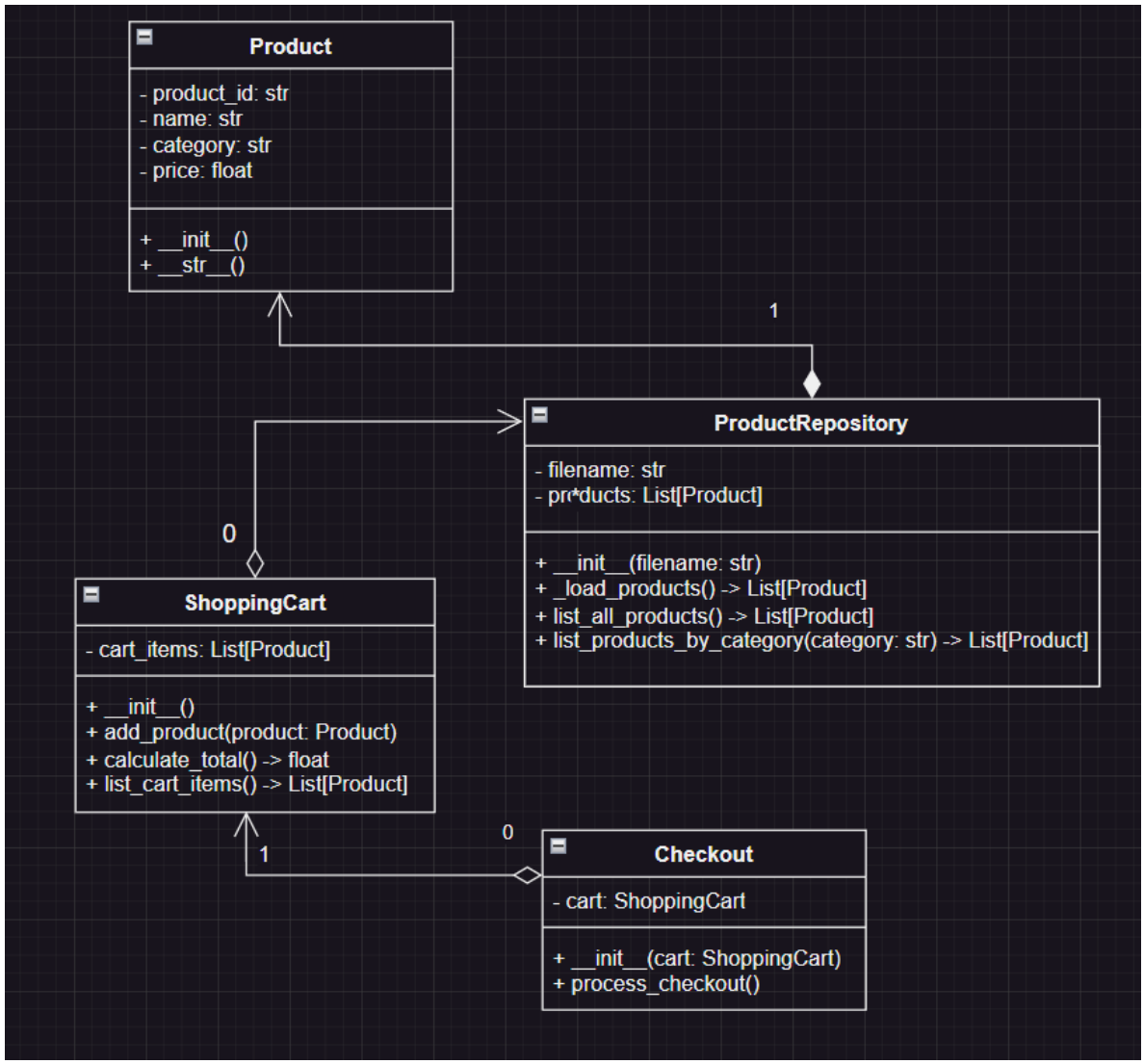
Activity diagram:



Sequence diagrams:



Class diagrams:


Product
- product_id: str
- name: str
- category: str
- price: float
+ __init__()
+ __str__()
1
ProductRepository
- filename: str
- pr*ducts: List[Product]
+ __init__(filename: str)
+ _load_products() -> List[Product]
+ list_all_products() -> List[Product]
+ list_products_by_category(category: str) -> List[Product]
0
ShoppingCart
- cart_items: List[Product]
+ __init__()
+ add_product(product: Product)
+ calculate_total() -> float
+ list_cart_items() -> List[Product]
1
0
Checkout
- cart: ShoppingCart
+ __init__(cart: ShoppingCart)
+ process_checkout()