Shopi report

Maiker Alejandro Hernández Archila-2023202002

Joan Sebastián Duran Pradilla-20232020019

1. **User Stories**

   The user stories we have used for our project are the following:

- **David Gomez:** As a new user (seller or buyer), I want to be able to register by providing my personal information (name, email, phone number), so I can start using the platform and access the buying or selling features.
- **Maria Patiño:** As a seller, I want to be able to create a product listing by specifying its category (e.g., car, phone, appliance), condition (new or used), price, and description.
- **Diego Hernández:** As a buyer, I want to be able to search for products using filters such as category, condition (new or used), and price.
- **Dayan Gutierrez:** As a buyer, I want to be able to add products to the cart and proceed to payment to purchase the items.
- **Santiago Mesa:** As a buyer, I want to access a purchase history where I can see all my previous transactions with details of each product.
- **Erick Quiñonez:** As a seller, I want to be able to manage my inventory, seeing which of my products have been sold and which are still available.
- **David Pulido:** As a seller, I want to be able to create a product listing by specifying its category (e.g., car, phone, appliance), condition (new or used), price, and description.
- **Santiago Sanchez:** As a seller, I want to receive a notification when someone buys one of my products, so I can prepare the shipment and communicate with the buyer if necessary.
- **Felipe García:** As a buyer, I want to be able to add products to a wishlist or favorites list, to save items I’m interested in and view or purchase them later without having to search for them again.

2. Object-Oriented Principles Analysis

- **Encapsulation:**

  The diagram displays encapsulation through the use of class attributes and methods. For example, classes like `User`, `Buyer`, and `Seller` each have specific attributes (`name`, `email address`, `balance`) and methods (like `add-product`, `remove-product`) that are relevant only to that class, hiding the internal state of objects.

- **Inheritance:**

  Inheritance is clearly represented by classes such as `Used Products` and `New Products`, which inherit common attributes (`warranty`, `manufacturer` for new products, and `condition`, `age` for used products) from their parent class `Product Factory`. Similarly, specific products such as `Used Cars` or `New Phones` inherit the basic structure of their respective product categories while adding their unique characteristics (e.g., `model`, `brand`, `number of cameras` for phones).

- **Polymorphism:**

  This principle could be inferred from the fact that different types of products (cars, phones, appliances) all derive from the `Product Factory`. Polymorphism allows the program to handle objects of these subclasses through a common interface or parent class (`Product Factory`) without needing to know their specific types at runtime.

- **Abstraction:**

  The diagram abstracts the functionality of certain processes, like payment (`Payment` class) and checkout (`Buyer` class), simplifying how these processes are represented without exposing the underlying complexities. The `Purchase Handler` class also abstracts the handling of the cart for adding and removing products.

- **Association and Aggregation:**

  There is a clear relationship between `User`, `Buyer`, and `Seller` with the `Product Factory`, `Payment`, and `Invoice` classes. The associations between `Buyer` and `Checkout`, as well as `Seller` and `Available Products`, depict a structure where objects communicate and work together to fulfill the responsibilities of the system. Aggregation is implied in classes like `Invoice`, which depends on a collection of products (`productList`).

- **Dependency:**

  The `Payment` class depends on the `Purchase Handler` and `Buyer` to process the payment, indicating that changes in one class could affect others. This dependency shows how different objects work in conjunction to complete operations such as payment processing.

3. CRC Cards (Class-Responsibility-Collaborator)

Seller	
Responsability	Collaboration
• Manage available products • Add Products • Retrieve and remove products	• Product Factory • User (inherits attributes from User)

Buyer	
Responsability	
• Perform purchases (checkout).	• Payment • Seller (for products) • User (inherits attributes from User)

Product Factory	
Responsability	Collaboration
• Create and manage products. • Retrieve and set product details.	• Purchase Handler • Seller • Buyer

Purchase Handler	
Responsability	Collaboration
• Manage the shopping cart (add and remove products).	• ProductFactory • Buyer

Payment	
Responsability	Collaboration
• Process payments. • Retrieve payment details.	• Buyer • Invoice

Invoice	
Responsability	Collaboration
• Generate invoices for purchases.	• Buyer • Payment

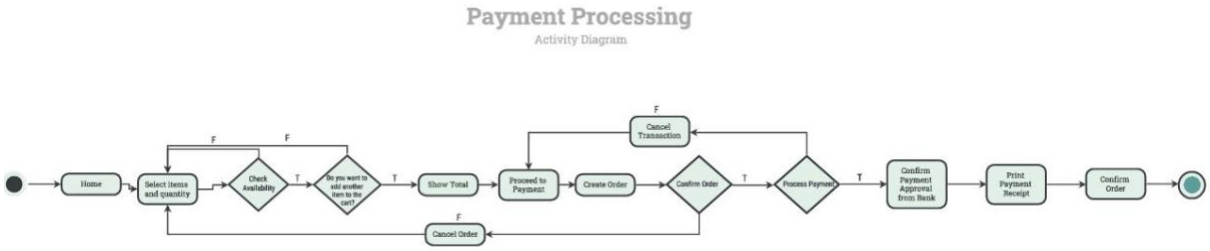
User	
Responsability	Collaboration
• Store basic user information (name, email, balance, phone number). • Set user information.	• Seller • Buyer

Used Products (inherits from ProductFactory)	
Responsability	Collaboration
• Manage used products, including condition and age.	• Seller • Buyer

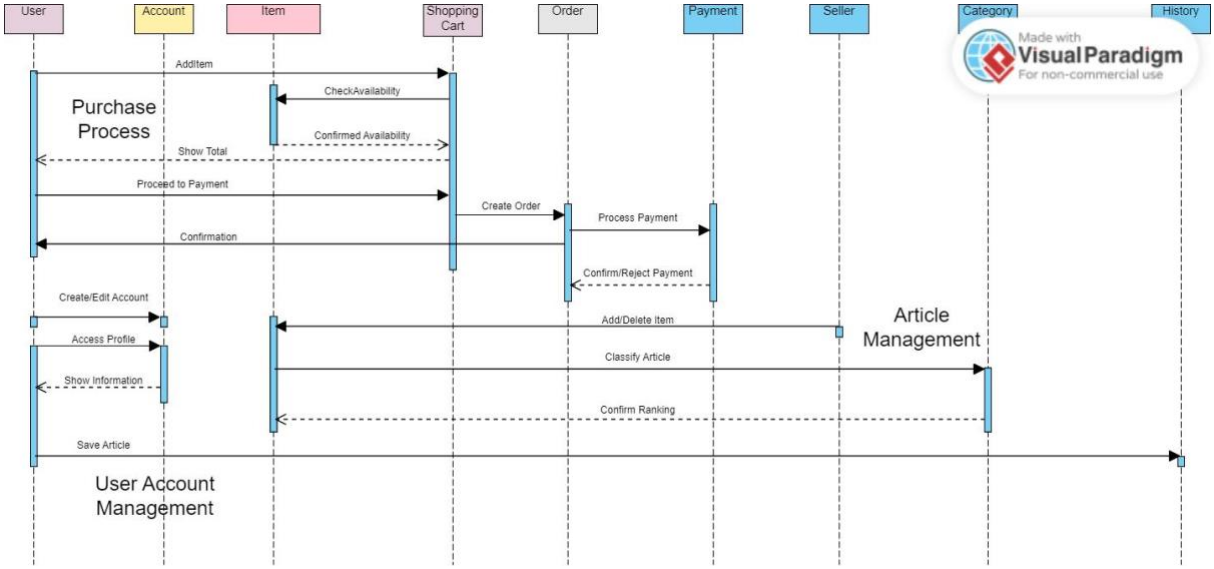
New Products (inherits from ProductFactory)	
Responsability	Collaboration
• Manage new products, including warranty and manufacturer.	• Seller • Buyer

User Factory	
Responsability	Collaboration
• Create users of type "Seller" or "Buyer" (user factory).	• Seller • Buyer

4. Activity Diagrams



5. Sequence diagram.



6. Class diagram