Ankara Yıldırım Beyazıt University
Department of Computer Engineering

CENG 201 – Object Oriented Programming
Course Project

G10: e-Commerce Site

Analysis Report

Aybüke Karaçavuş
Sema Nur Yılmaz
Sıla Dolaş
Murathan Karakaya

Instructor: Muhammed Abdullah Bülbül
Teaching Assistant: Elif Şanlıalp, Yusuf Şevki Günaydın
Date: 28/10/2024

Table of Contents

1. Introduction

This report presents the development of a shopping cart system for an e-commerce platform. The system allows users to browse products, add items to their cart, and proceed through a checkout process. Several key software design patterns have been implemented to enhance functionality and maintainability.

The main objectives of this project are to demonstrate effective product management, support various discount strategies, and provide dynamic features for the products. The following features are highlighted in this report:

- ***Product Management**: Users can add and remove products from their cart while the system calculates the total price dynamically.*
- ***Checkout Process**: Various discounts and promotions, such as percentage discounts and buy-one-get-one offers, are applied at checkout using the Strategy Pattern.*
- ***Observer Pattern**: This pattern is used to notify users when products are added or removed from the cart, ensuring the system stays updated in real-time.*
- ***Strategy Pattern**: Multiple discount strategies are integrated, allowing flexibility in how discounts are applied during the checkout process.*
- ***Decorator Pattern**: The system allows the dynamic addition of features to products, such as gift wrapping, without modifying the original product class.*

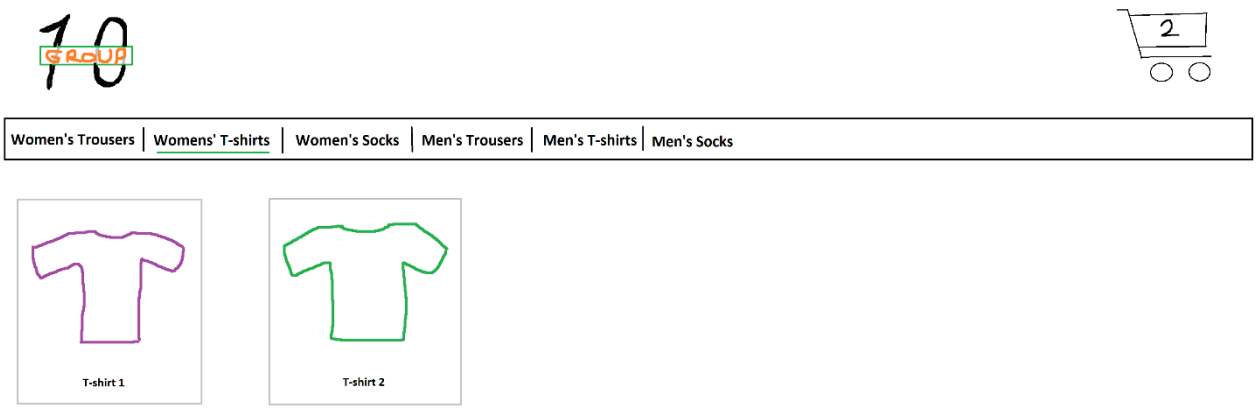

Figure 1. Mock-up drawing of site

2. Requirements

2.1. Functional Requirements

The functional requirements for the system include:

User Account Management

- Users should be able to create accounts, log in, and manage their profiles.
- Users should be able to change their password.

Product Browsing and Viewing

- Users should be able to browse and view products available in the system.
- Users should be able to sort products by price, either from lowest to highest or from highest to lowest.

Shopping Cart Management

- Users should be able to add, remove, or modify products in their cart.
- Users should be able to view the total amount of their cart, which should automatically update as products are added or removed.
- Free shipping should be available if the cart total exceeds a certain amount.
- Users should be able to choose the type of shipping (e.g., faster but more expensive option).

Checkout and Payment Process

- Users should have access to various discount options during checkout, such as percentage discounts, buy-one-get-one-free, and coupon discounts. The system supports different discount strategies, applied dynamically during checkout using the Strategy Pattern.
- Users should receive a confirmation notification once the payment is completed.
- Upon purchase completion, the stock levels of products should decrease, with a low or out-of-stock warning appearing when necessary.

Notifications System

- Users should receive notifications when products are added to or removed from their cart. The system uses the Observer Pattern to notify users. Users should receive a payment confirmation notification after completing the purchase.

Product Customization

- Users can apply additional features (e.g., gift wrapping) to products via the Decorator Pattern without modifying the core product structure.

Order Management

- Users should be able to view their order history.

Product Ratings and Reviews

- Users should be able to rate products that they have purchased.

Seller Management

- Sellers should be able to add and update products in the system.
- Seller can select the category of the product and specify the amount of stock.

2.2. Non-Functional Requirements

The non-functional requirements for the system include:

- **Performance:**
    - The system should respond to user actions, such as adding or removing items from the cart, quickly.
    - The system should be able to handle at least 100 concurrent users browsing products and adding items to the cart without significant slowdowns.
- **Reliability:**

    - The system should maintain high availability, ensuring users can access the platform consistently without unplanned interruptions.
- **Scalability:**
    - The platform should support the addition of new products, users, and promotions without requiring significant system changes.
    - The architecture should allow easy addition of new features and products.
- **Security:**
    - User data, including personal information and payment details, should be encrypted and protected.
- **Usability:**
    - The system should be user-friendly, with intuitive navigation to allow users to easily find and manage products.
    - The system should offer an intuitive user interface, ensuring that users can easily navigate, browse products, and complete purchases without the need for technical assistance.
- **Maintainability:**
    - The system architecture should be modular and maintainable, allowing easy updates or modifications without affecting the overall functionality.

3. System Models

3.1. Scenarios

Scenario 1: Ali enters the site. Looks for a dress for his sisters birthday. He finds one, adds to the cart, proceeds to check out. He enters the discount code that his company gave. Gets 20 percent discount. He also selects gift wrapping option for his present. He then completes the payment and exit.

Scenario 2: Veli enters the site. He first adds 2 t-shirt to the cart but after he sees buy one get one campaign for socks he decides to remove one t-shirt and adds the 2 socks. Site notifies him for the added/removed items. He sees the discount for the socks at payment. He then completes the payment and exit.

Scenario 3: I opened the site and created a new user registration. I was in search of a sweater, filtered through clothes. After reviewing the rating of a sweater that caught my attention, I added it to my cart. I confirmed my cart, then the shipping fee was added. I entered my address and card information and saved it for later. After making the payment, I received a notification that the payment was confirmed.

Scenario 4: I logged in to the site with my information to repurchase a book I had previously purchased for myself. I added the book to my cart by viewing my old orders. I had a discount coupon and entered the code. I added a gift package for my friend. Since my address and payment card were registered, I completed my shopping quickly.

3.2. Use Cases

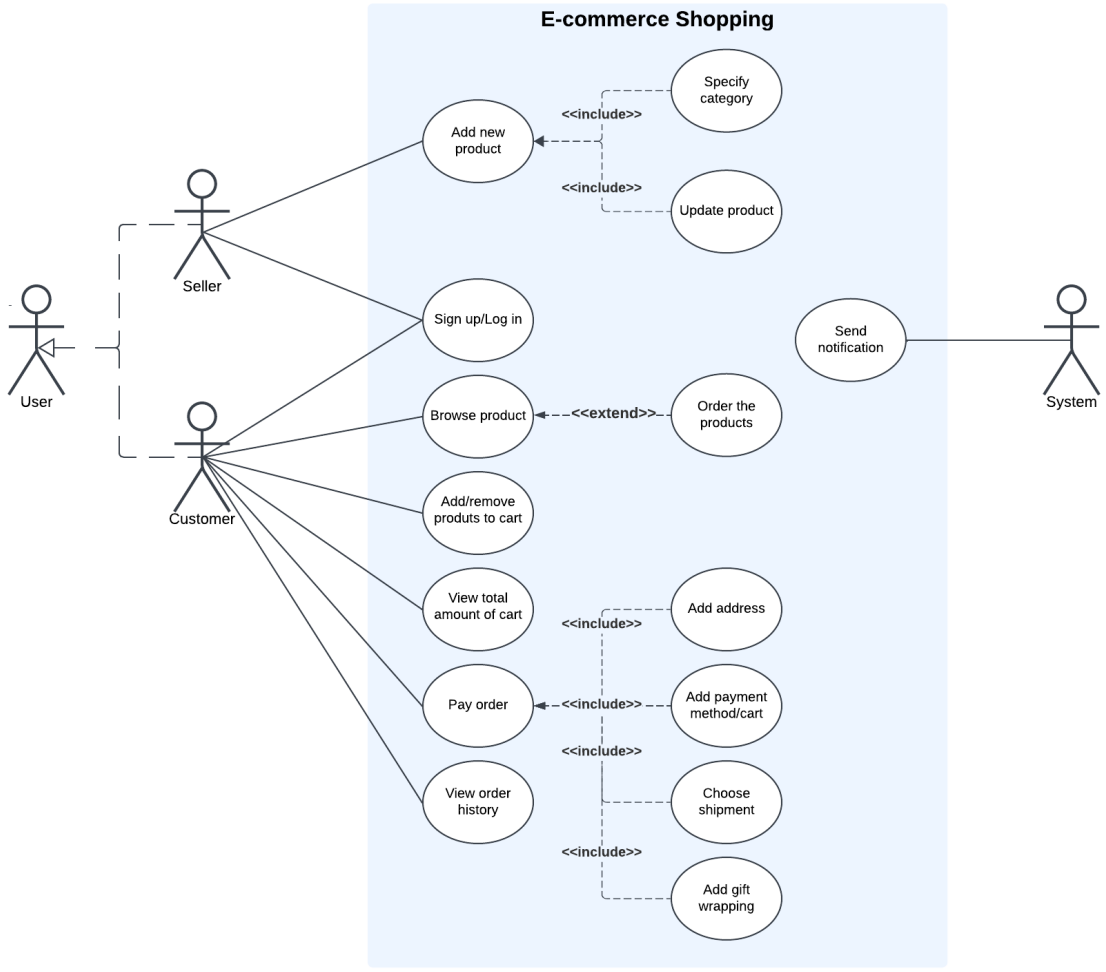

Figure 2. UML use case diagram

User

- **Sign up/Log in:** User must create an account to enter the site. If they already signed up to the site they can just log in.

Seller:

- **Add new product:** Seller can add new product to their store. While adding the product, they must choose a category and determine the stock. Later that seller can update product's informations.

Customer:

- **Browse product:** Customer can navigate through the categories and search for a product by name that they want. Also they will be able to order the products as price as low to high or high to low.

- **Add/remove product to cart:** Customer can add or remove some products to the cart as they wish.
- **View total amount of cart:** When customer finish their shopping, they can see the total cost for the products before discount.
- **Pay order:** In the payment stage, if the customer has not added their address before, they must add it. IF already entered, no need to re-enter the address again. Same for the payment card. They can add new card or use previous card's informations. Also there will be shipment options such as faster delivery with more charge or normal delivery time with less charge. Additionally, customer can add gift wrapping option for their order. Finally, if there is available any discount option, they can apply it.
- **View order history:** Customers will be able to view their old orders along with its prices.

System

- **Send notification:** System notifies the user an item successfully added to the cart or removed from the cart.

3.3. Object and Class Model

- **User**
  - **Seller/Admin**
  - **Customer**
- **Customer**
  - Can have many orders
  - Has a shopping cart
  - Has a payment method
- **Seller**
  - Has products
- **Product**
  - Belongs to many categories
  - Can belong to one Cart or one order
- **Cart**
  - Belongs to a user
  - Can contain multiple items
- **Order**
  - Belongs to a user
  - Can contain multiple items
- **Category**
  - Has many products
- **Shipping**
  - Belongs to an order
- **Payment Method**
  - Belongs to a user

- **System**
    - send notifications to the user
-
- **Wishlist**
- **Order History**

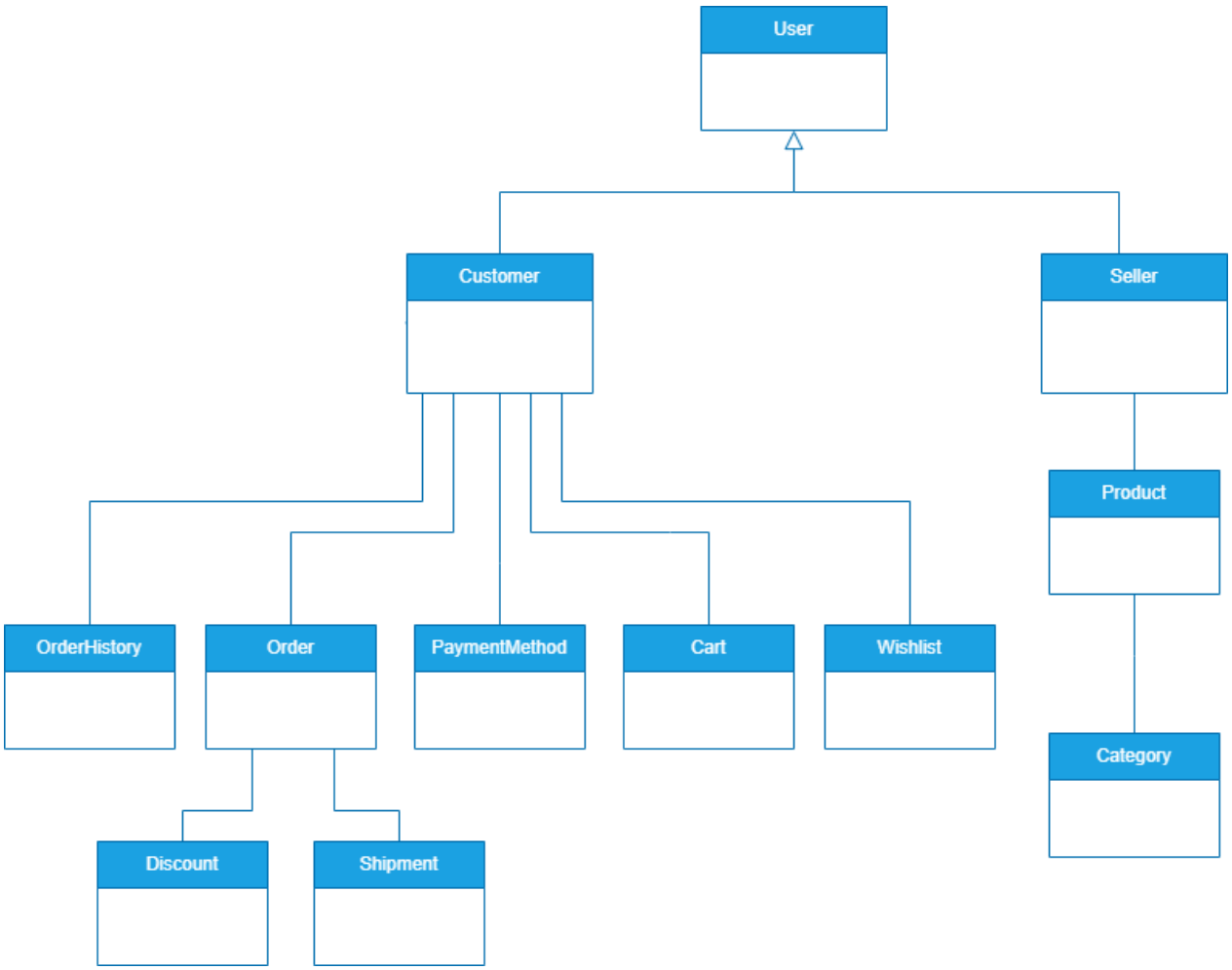

Figure 3. UML Class Diagram

3.4. User Interfaces

Figure 4. Item added to the cart

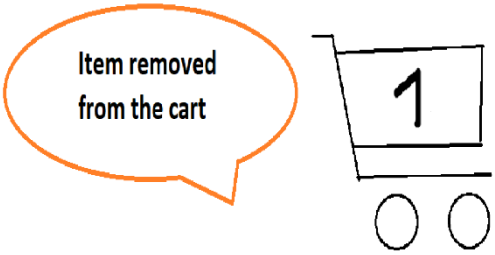

Figure 5. Item removed from the cart

Figure 6. Item selected

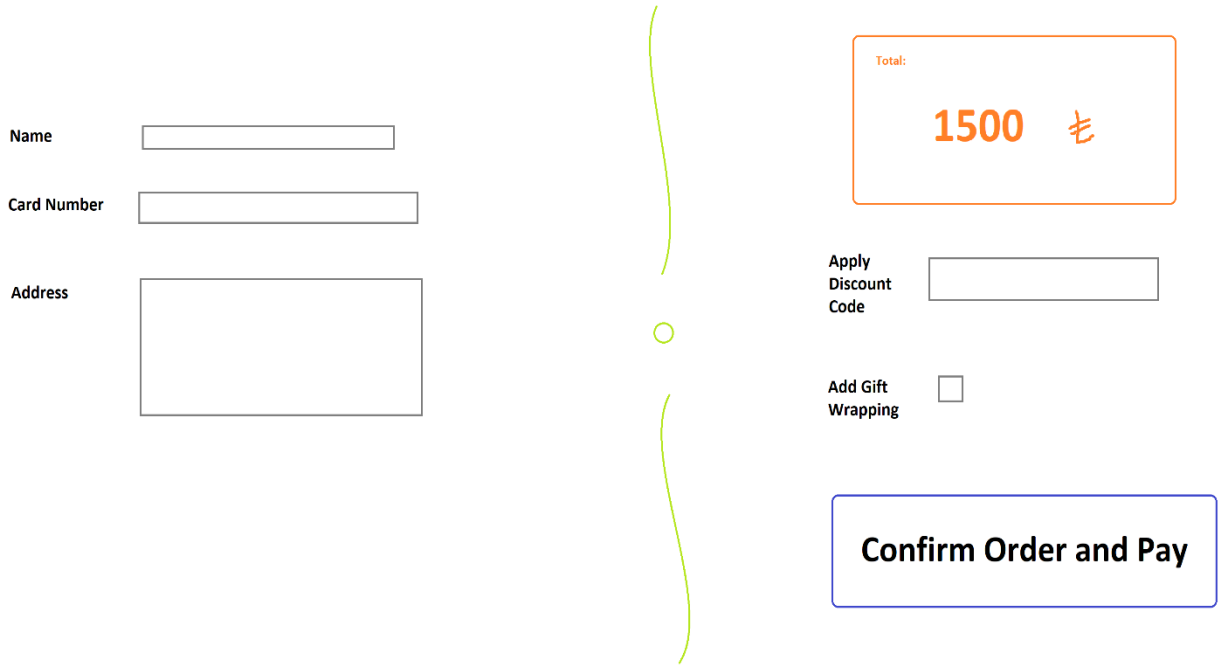

Figure 6. Checkout

4. Conclusion

In this report, we identified requirements of our project. Created scenarios and use cases, discussed on classes and showed some mockup images. We also discussed how to proceed with the project and how can we improve. All members worked together throughout the project and expressed their opinions.