Fatima Aishalyn Pepino
CS350 Software Engineering
Professor Thomas Chu
May 15, 2023

The Problem

Selene Clothing is a clothing store that sells clothes in various colors and sizes. The owner of Selene Clothing reached out to you to help them create a website where a customer can order items online in order to avoid the busyness of the mall.

The program must:

- The system should allow customers to create, sign in, and manage their accounts.
- The system should allow the customers to search for items by name, description, or category.
- The system should allow the customers to add, remove, and modify their carts.
- The system should allow customers to choose sizes and colors before adding items to the cart.
- The system should keep a record of all customer purchase transactions.
- The system should check the availability of the items.

REQUIREMENTS

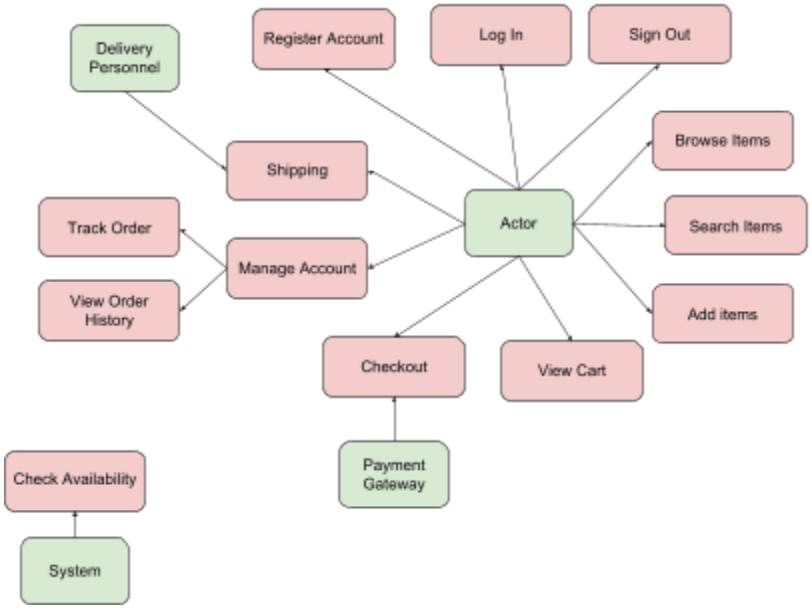
Business Model

Selene Clothing is a clothing retailer offering a range of options from different types of tops, pants, shorts, skirts, swimwear, athletic wear, and accessories. The clothes are made to cater to teenagers and young adults.

A) Selene Clothing is launching a website where customers can shop for clothes without having to go to the mall. Its whole purpose is to provide services for customers who wish to buy clothes online and to look at the clothing store's catalog before going to the store (for some people who wish to try on the clothes first before buying).
B) The clothing brand has its own group of designers and graphic designers to come up with unique designs and styles. The company releases a new set of clothes every season, although it also sells basic clothes for layering or for casual styling all throughout the year. The basics are permanent in the brand's catalog. It will get restocked no matter the season.
C) The online store does not allow returns and cancellations at the moment.
D) A third-party provider will authorize the payment.
E) Selene Clothing keeps its inventory at a physical warehouse. After the customer's order has been placed, the details of the order will be sent to the database that is used in the warehouse. This will generate the shipping label.
F) Once the order process is complete, the shipping label will be placed on the box. A third-party company will be shipping out the packages.

First UML Diagram

Second UML Diagram



How would you ensure the software is reusable and with as less maintenance as possible?

Because an online store has a lot of functionalities for it to work properly, data coupling is best to use in order to ensure the website's reusability. Data coupling allows each function to have its independence so that when one module is under maintenance, the other modules will not be affected by this change, or at least the changes would have minimal impact on them. And because data coupling provides flexibility, this helps with the easy accommodation of new changes in the program. The flexibility also helps when new requirements arise. Data coupling also supports integration with third-party services which the website needs as it is partnering with a third-party provider to help with the payments done online.

GLOSSARY (words to keep in mind)

- Register - refers to the act of recording an event, transaction, name, or other information, or aggregation of stored data, usually containing past events, transactions, names, or other information.
- Log In - authenticate to a user interface by providing your credentials.
- Sign Out - go through the procedures to conclude the use of a computer system, database, or website.
- Browse - survey items for sale in a leisurely and casual way.
- Item - an individual article or unit, especially one that is part of a list, collection, or set.
- Checkout - a point at which goods are paid for in a supermarket or other stores (physical or online).
- Shipping update - shows the time and place that the package has arrived at a specific location.
- Tracking status - shows whether the package is yet to be processed, has been processed, is in transit, or has been delivered.
- Order: An order can be modeled as an object that represents a customer's purchase.

USE CASES

Register Account

Brief Description
The *Register Account* use case allows the customer to create an account by providing their personal information like name, and email address, and creating their own password for easier checkout.

Step-by-Step Description
1. The customer opens the online store.
2. The customer clicks *Create account.*
3. The customer inputs their name, email address, and password they have in mind in the respective text boxes.
    a) The password should be 8 characters long, including one capital letter, one number, and one special character.
4. The system shows that their account has been created.

Log In

Brief Description
The *Log In* use case allows the customer to access their existing account.

Step-by-Step Description
1. The customer opens the online store.
2. The customer clicks the *Log In* button.
3. The customer types their email address and password in the text boxes provided.
4. If the email address and password match, the customer has accessed their account.

Sign Out

Brief Description
The *Sign Out* use case allows the customer to get out of their account.

Step-by-Step Description
1. The customer simply clicks the *Sign Out* button.
2. The system asks if they are sure to do so.
    a) If yes, they will be logged out of their account and will be back to the site's main page.
    b) If no, the customer will remain where they were.

Browse Items

Brief Description The *Browse Items* use case allows the customer to view the store's collection of items.
Step-by-Step Description 1. The customer opens the online store. 2. The system shows the items' names, images, descriptions, and prices. 3. The customer skims through the online store to look for what they want.

Search Items

Brief Description The *Search Items* use case allows the customer to search for items by keyword, category, brand, or other relevant attributes.
Step-by-Step Description 1. The customer searches for an item via the search box. 2. The system shows the categories of the items by color, size, price range, and according to what type of clothing it is: top, bottoms, etc. 3. The system returns relevant items that match the word the customer searched for. 4. The system shows the items' names, images, descriptions, and prices.

Add to Cart

Brief Description The *Add to Cart* use case allows the customer to add items to their shopping cart for purchase.
Step-by-Step Description 1. The customer views the store's collection of items. 2. The customer sees an item they want to buy. 3. The customer clicks on the item. 4. The system shows more detailed information about the item. 5. The customer adds the item to their cart.

View Cart

Brief Description The *View Cart* use case allows the customer to view the contents of their shopping cart, including the list of items added, the total price, and the option to modify or remove items.

Step-by-Step Description

1. The customer clicks the *View Cart* button.
2. The system returns the list of items added and their prices, and the total price.
3. The customer can remove items they do not want to purchase anymore.
4. The customer can also modify the number of a specific item they wish to purchase.
5. The system returns the updated list, the prices, and the total price.

Checkout

Brief Description
The *Checkout* use case allows the customer to proceed to checkout and provide shipping and payment information to complete the purchase.

Step-by-Step Description

1. The customer clicks the *Checkout* button.
2. The system asks the customer to put their name and address for shipping information, and payment information.
3. The bottom part of the screen shows the *Place Order* button, which the customer clicks once they are ready.

Manage Account

Brief Description
The *Manage Account* use case allows the customer to manage their account information, including adding and/or removing personal information, shipping addresses, payment methods, and password.

Step-by-Step Description

1. The customer clicks the *Manage Account* button.
2. The system shows which information you want to add, modify, or change.
    a) If a customer wants to change their password, they can click the *Password* button.
    b) The system inputs the password they have in mind.
3. The system shows that their information has been changed.

Shipping

Brief Description
The *Shipping* use case allows the customer to choose which shipping method they would like.

Step-by-Step Description

1. The customer clicks the *Checkout* button.
2. The system shows different options how a package can be shipped, whether they want standard shipping (which takes 7 business days), overnight shipping, or express shipping (which takes 3 business days or less).
3. The system calculates the shipping costs which will reflect on the total price.
4. The system uses the information put here in generating a label for the database, which is for employees.

View Order History

Brief Description
The *View Order History* use case allows the customer to view their order history, including the list of new orders, past orders, their status, and order details.

Step-by-Step Description
1. The customer clicks the *View Order History* button.
2. The system shows the customer's order history.
3. The customer chooses which order they want to check or track and clicks it.
4. The system shows the customer's order details, which include the breakdown of the order, its total price, and its tracking status.

Track Order

Brief Description
The *Track Order* use case allows the customer to track the status of their orders, including shipping updates and delivery estimates.

Step-by-Step Description
1. The customer clicks the *Track Order* button.
2. The system shows the order's shipping updates.
3. The system shows when the package is expected to be delivered.

Check Availability

Brief Description
The *Check Availability* use case allows the system to check whether an item is available or sold out.

Step-by-Step Description
1. The customer chooses an item.
2. The system automatically shows the availability of the item.

a) If the customer picked a color first, the sizes of which the item is not stocked will be greyed out.
b) If the customer chose their size first, the color of which the item is not stocked will be greyed out.

ANALYSIS

Functional model

- A new customer checks out their shopping cart
    - ➔ The customer opens the website of the store.
    - ➔ The customer registers an account by putting in their name, email address, and password.
    - ➔ The customer browses through the main page for clothing.
    - ➔ The customer sees an item they like, picks a color, chooses their size, then adds the item to the cart.
    - ➔ The customer clicks the *Checkout* button.
    - ➔ The system will take the customer to the checkout process where the customer is expected to write down their name, phone number, email address, and shipping address.
    - ➔ The customer will choose their preferred shipping method.
    - ➔ The customer will be asked by the system to provide payment information and a billing address.
    - ➔ Once the customer is done putting in their information, the customer clicks the *Checkout* button again.
    - ➔ The system will ask the customer whether they are sure or they want to cancel.
    - ➔ If the customer chooses “yes”, they will see the order details on the screen and the order confirmation in their email. If not, the customer’s screen will stay on which page it left.

- An existing customer wants to check their order's tracking status
    - ➔ The customer logs into their account using their email and password.
    - ➔ The customer goes to *Manage Account*.
    - ➔ The customer clicks *View Order History*.
    - ➔ The customer chooses which order they want to check.
    - ➔ The system returns the order details which is also where a *Track Order* button can be seen.
    - ➔ The customer clicks the *Track Order* button.
    - ➔ The system shows the tracking status of the order, including updates of its location and the expected delivery date.

- An existing customer wants to remove an item from their shopping cart
    - ➔ The customer logs into their account using their email and password.
    - ➔ If the customer has already an existing shopping cart from a previous visit, the customer can go to *View Cart*.
    - ➔ In *View Cart*, they can remove an item by clicking the *Remove* button.
    - ➔ The system will remove the item from the shopping cart.

- An existing customer wants to change their password
    - ➔ The customer logs into their account using their email and password.
    - ➔ The customer goes to *Manage Account*.
    - ➔ The customer chooses *Change Password*.
    - ➔ The customer types in their previous password for verification.

- ➔ The customer types their new password that has to meet the password requirements.
- ➔ The system returns “Password changed successfully” if the previous password the customer typed down matches their current password and if the new password meets the requirements.

Noun Extraction

The **Online Clothing Store** allows the **customer** to order clothes online. They can register an **account**, log into an existing account, and sign out from their account. They can browse **items** at their own pace if they have no idea what to buy. They can also search for specific **items** that they want. The **customer** can categorize the browsing **page** according to what they want like the **type of clothing**, the **size**, the **color**, or the **price range** according to their **budget**. The **customer** can also add, remove, and modify their **cart**. They can check out their shopping cart whenever they are ready. The **customer** can also manage their own **account** by adding, removing, and modifying personal information, password, address, and **payment information**. The customer can check their **order history** and **order details**, and track their order's **status**, which the **system** can provide the information for them. The **system** automatically checks the **inventory** of an item's **availability** whenever an item gets checked out in-store or online.

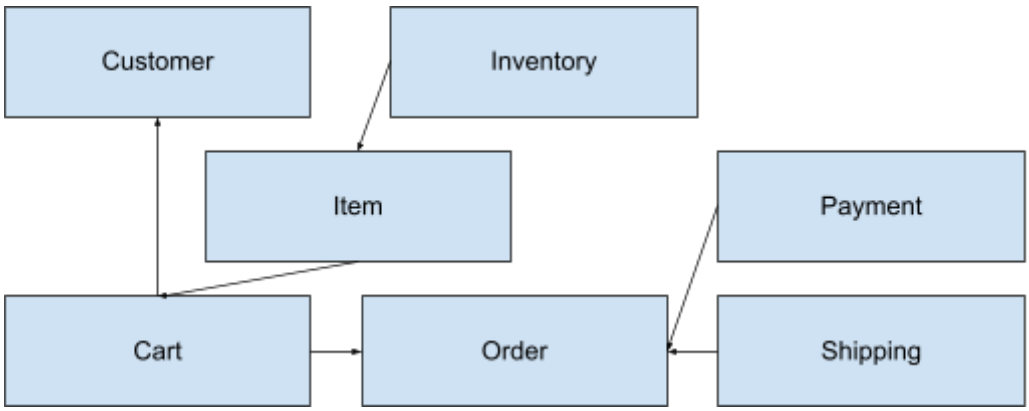
Nouns: Online Clothing Store, customer, account, items, page, type of clothing, size, color, price range, budget, cart, account, payment, information, order, history, details, status, system

Exclude: Online Clothing Store, account, page, type of clothing, size, color, price range, budget, account, information, history, details, status, system

Candidate Classes: customer, items, cart, order, payment, shipping, inventory

Class Diagram

First Iteration

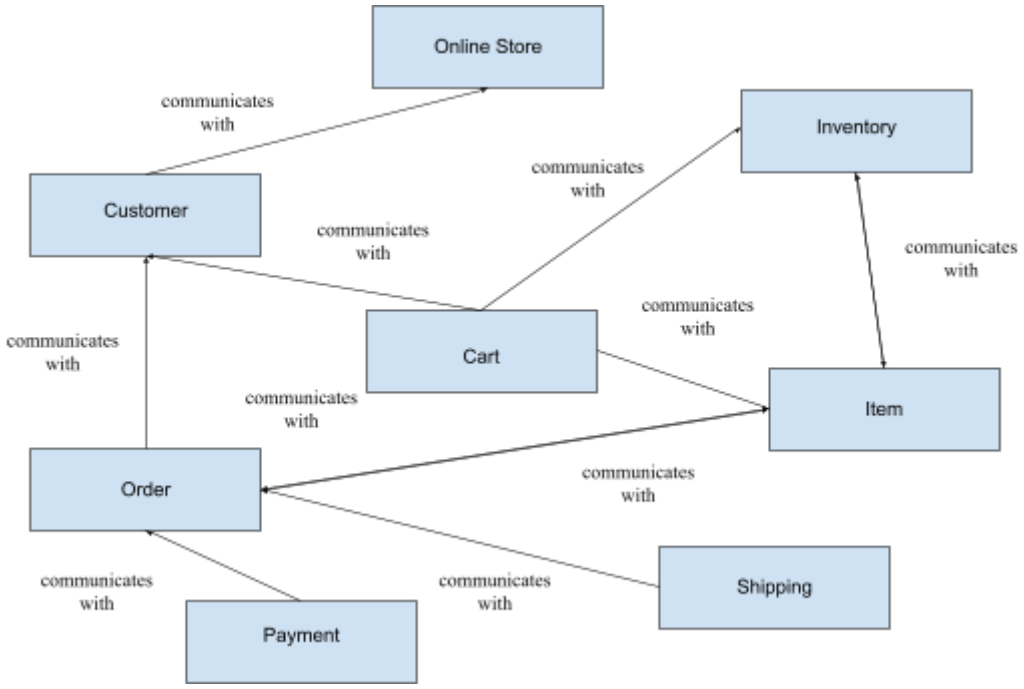


Attributes

Customer (Entity)	Inventory (Entity)	Item (Entity)	Cart (Entity)
customerName customerEmail customerAddress, customerPaymentInfo rmation	itemAvailability itemQuantity	itemName itemNumber itemDescription itemPrice itemImage itemSize itemColor	cartNumber associateCustomer listOfItemsInCart

Order (Entity)	Shipping (Entity)	Payment (Entity)
orderNumber customerInformation orderItems totalPrice shippingDetails	shippingNumber shippingAddress shippingMethod trackingInformation	paymentID paymentMethod Amount

Second Iteration



Class Responsibility Cards

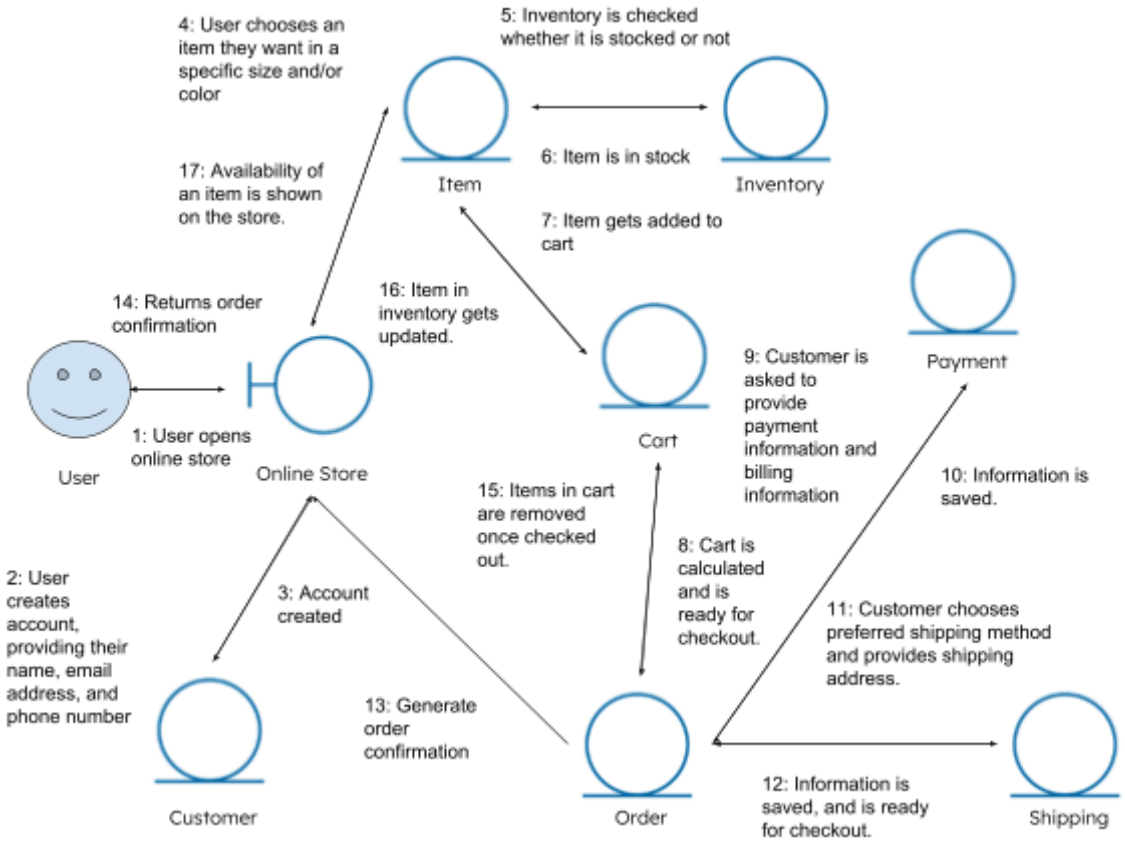
Item Class	**Inventory Class**
Responsibilities 1. Retrieve and show the item name 2. Retrieve and show item number 3. Show item description 4. Show item price 5. Show item image 6. Show item sizes 7. Show item colors	**Responsibilities** 1. Show how many items are left in stock 2. Show which items are sold out. 3. Show which items will be restocked and when it gets restocked 4. Removes discontinued items from the store page 5. Update availability on store page
Collaboration(s) 1. Inventory Class 2. Order Class	**Collaborations(s)** 1. Item Class

Cart Class	**Order Class**
Responsibilities 1. Adds items to the cart 2. Updates quantity of the items in the cart 3. Removes items 4. Calculates subtotal 5. Provides cart summary 6. Handles checkout process 7. Saves cart data	**Responsibilities** 1. Creates order by asking for relevant information such as customer's personal information, shipping address, payment information, and list of items in the order 2. Calculates total price 3. Updates inventory after an order is placed 4. Initiates payment transaction, validate payment details, and handle errors that may occur during the process 5. Tracks and updates order status like *pending, shipped,* and *delivered.* 6. Provides order history
Collaboration(s) 1. Item Class 2. Inventory Class	**Collaboration(s)** 1. Customer Class 2. Item Class

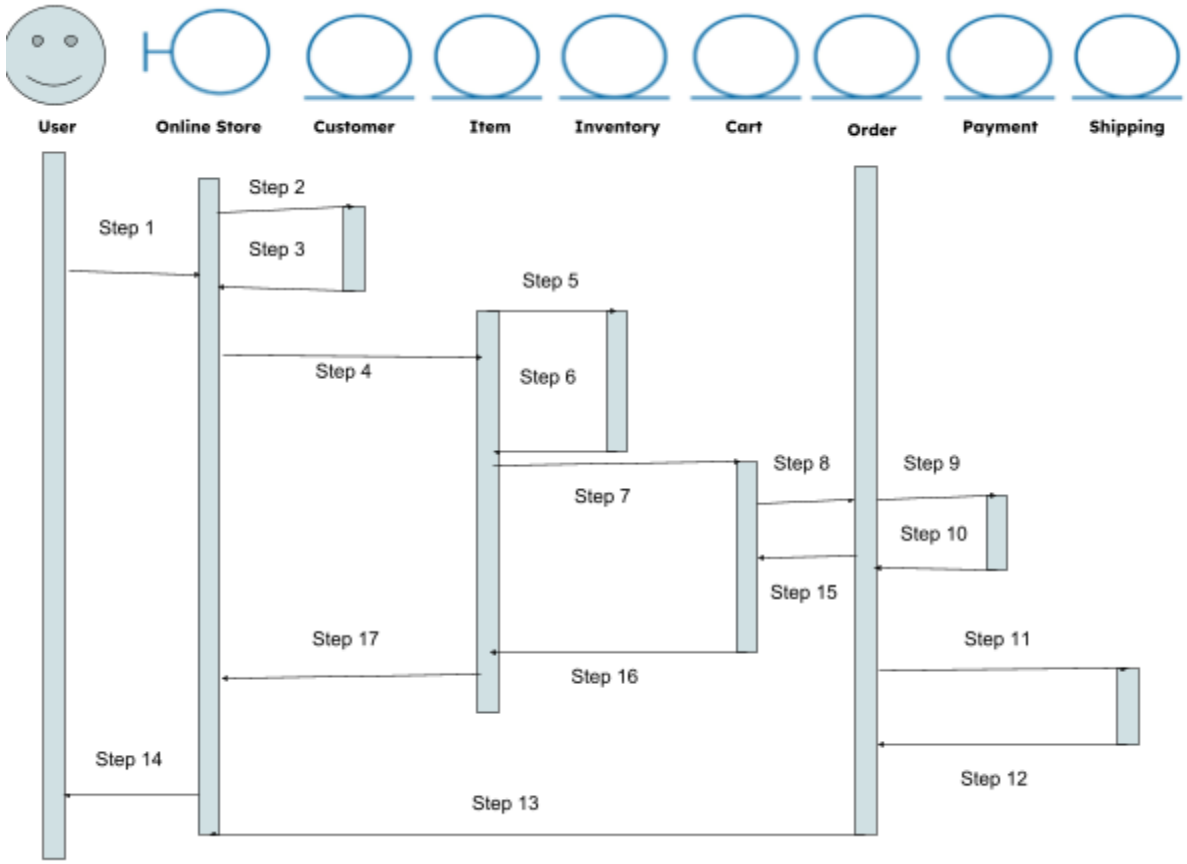
Shipping Class	**Payment Class**
Responsibilities 1. Calculates shipping cost 2. Tracks shipment status and location 3. Handles shipping methods 4. Calculates estimated delivery date based on the shipping method and location 5. Communicate shipping updates such as *shipped, in-transit, out for delivery,* or *delivered* notifications to customer	**Responsibilities** 1. Processes customer payments 2. Validate payment details such as credit card information and billing address 3. Obtains authorization from the payment provider to charge the customer 4. Ensures that the captured payment is accurate and matches the order total 5. Generates payment receipts including transaction ID, payment method, payment amount, etc.
Collaboration(s) 1. Order Class	**Collaborations(s)** 1. Order Class

Customer Class
Responsibilities 1. Creates customer account 2. Manages customer profile 3. Authenticate customer log-in 4. Sign out of account 5. View order history
Collaboration(s) 1. Order Class 2. Cart Class 3. Payment Class

Communications Diagram



Sequence Diagram



DESIGN

Pseudocode

```
items = []

users = []


function addItem(name, price, quantity):
    item = {
        "name": name,
        "price": price,
        "quantity": quantity
    }
    items.append(item)


function registerUser(name, email,
password):
    user = {
        "name": name,
        "email": email,
        "password": password,
        "cart": []
    }
    users.append(user)
```

```
function loginUser(email, password):
    for user in users:
        if user["email"] == email and
user["password"] == password:
            return user
    return null

function addToCart(user, item):
    user["cart"].append(item)

function removeFromCart(user, item):
    user["cart"].remove(item)

function calculateCartTotal(user):
    total = 0
    for item in user["cart"]:
        total += item["price"]
    return total
```

```
if user != null:
    item1 = items[0]
    item2 = items[1]
    addToCart(user, item1)
    addToCart(user, item2)
    removeItem(user, item1)

    cartTotal = calculateCartTotal(user)
    print("Cart total:", cartTotal)
else:
    print("Invalid login credentials"
```

JAVA CODE

```
import java.util.ArrayList;
import java.util.List;

public class Main {

    public static void main(String[] args) {
        OnlineStore store = new OnlineStore();

        store.addItem("Item 1", 10.99, 20);
        store.addItem("Item 2", 5.99, 15);

        store.registerUser("Aisha Pepino",
"aisha@example.com", "password123");

        User user =
store.loginUser("aisha@example.com",
"password123");
        if (user != null) {
            Item item1 = store.getItems().get(0);
            Item item2 = store.getItems().get(1);

            store.addToCart(user, item1);
            store.addToCart(user, item2);
            //store.removeFromCart(user, item1);

            double cartTotal =
store.calculateCartTotal(user);
            System.out.println("Cart total: " +
cartTotal);
        } else {
            System.out.println("Invalid login
credentials");
        }
    }

    static class Item {
        private String name;
        private double price;
        private int quantity;

        public Item(String name, double price,
int quantity) {
            this.name = name;
            this.price = price;
            this.quantity = quantity;
        }

        public void setName(String name) {
            this.name = name;
        }

        public String getName() {
            return name;
        }

        public void setPrice(double price) {
            this.price = price;
        }

        public double getPrice() {
```

```
        return price;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}

static class User {
    private String name;
    private String email;
    private String password;
    private List<Item> cart;

    public User(String name, String email,
String password) {
        this.name = name;
        this.email = email;
        this.password = password;
        this.cart = new ArrayList<>();
    }

    public String getName() {
        return name;
    }

    public String getPassword() {
        return password;
    }

    public String getEmail() {
        return email;
    }

    public List<Item> getCart() {
        return cart;
    }

    public void setCart(List<Item> cart) {
        this.cart = cart;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setPassword(String
password) {
        this.password = password;
    }
}

static class OnlineStore {
    private List<Item> items;
```

```
    private List<User> users;

    public OnlineStore() {
        this.items = new ArrayList<>();
        this.users = new ArrayList<>();
    }

    public void addProduct(String name,
double price, int quantity) {
        Item item = new Item(name, price,
quantity);
        items.add(item);
    }

    public void registerUser(String name,
String email, String password) {
        User user = new User(name, email,
password);
        users.add(user);
    }

    public User loginUser(String email,
String password) {
        for (User user : users) {
            if (user.getEmail().equals(email)
&& user.getPassword().equals(password)) {
                return user;
            }
        }
        return null;
    }

    public void addToCart(User user, Item
item) {
        user.getCart().add(item);
    }

    public void removeFromCart(User
user, Item item) {
        user.getCart().remove(item);
    }

    public double calculateCartTotal(User
user) {
        double total = 0;
        for (Item item : user.getCart()) {
            total += item.getPrice();
        }
        return total;
    }

    public List<Item> getItems() {
        return items;
    }

    public void setItems(List<Item> item)
{
        this.items = items;
    }
```