1. UML Use Case ：

（1）chatGPT

Question 1：

Identify actors and use cases for the system described below:

Consider the following problem description: A mail-order company wants to automate its order processing. The initial version of the order processing system should be accessible to customers via the web. Customers can also call the company by phone and interact with the system via a customer representative. It is highly likely that the company will enhance this system in upcoming years with new features. The system allows customers to place orders, check the status of their orders, cancel an existing order and request a catalog. Customers may also return a product but this is only possible through the phone, not available on the web. When placing an order, the customer identifies himself by means of customer number (only for existing registered customers) or by means of his name and address. He then selects a number of products by giving the product number or by selecting products from the online catalogue. For each product, information such as price, a description and a picture (only on demand as they are usually high-resolution images of large size) are presented to the customer. Also, the availability of the product is obtained from the inventory. The customer indicates whether he wants to buy the product and in what quantity. When all desired products have been selected, the customer provides a shipping address and a credit card number and a billing address (if different from the shipping address). Then an overview of the ordered products and the total cost are presented. If the customer approves, the order is submitted. Credit card number, billing address and a specification of the cost of the order are used on the invoice, which is forwarded to the accounting system (an existing software module). Orders are forwarded to the shipping company, where they are filled and shipped. Customers who spent over a certain amount within the past year are promoted to be gold customers. Gold customers have additional rights such as being able to return products in an extended time period as well as earning more bonus points with each purchase. In addition, in cases where a product is on back order, gold customers have the option to sign up for an email notification for when the particular product becomes available.

Based on the system description provided, the following are the actors and use cases:

Actors:

Customers (both registered and non-registered)

Customer Representatives

Shipping Company

Accounting System

Use cases:

1. Place Order: Customers can place orders through the web by selecting products, providing shipping and billing information, and paying with a credit card. They can also place orders by phone with the help of a customer representative.
2. Check Order Status: Customers can check the status of their orders by accessing the system either through the web or by phone with the help of a customer representative.
3. Cancel Order: Customers can cancel an order either through the web or by phone with the help of a customer representative.
4. Request Catalog: Customers can request a catalog through the web by filling out a form or by phone with the help of a customer representative.
5. Return Product: Customers can return a product by phone with the help of a customer representative.
6. Promote to Gold Customer: The system automatically promotes customers who spent over a certain amount within the past year to gold customers.
7. Extended Return Time Period: Gold Customers have an extended time period to return products.
8. Bonus Points: Gold Customers earn more bonus points with each purchase.
9. Sign Up for Email Notification: Gold Customers have the option to sign up for an email notification for when a product on back order becomes available.
10. Fill and Ship Orders: The shipping company fills and ships orders received from the system.
11. Forward Invoice to Accounting System: The system forwards the invoice containing credit card number, billing address, and the cost of the order to the accounting system for processing.

Question 2：

in the use case diagram, which actor should these use case link to

In the use case diagram, each use case should be linked to the actor who initiates or interacts with the use case. Based on the system description provided earlier, here are the actors that should be linked to each use case:

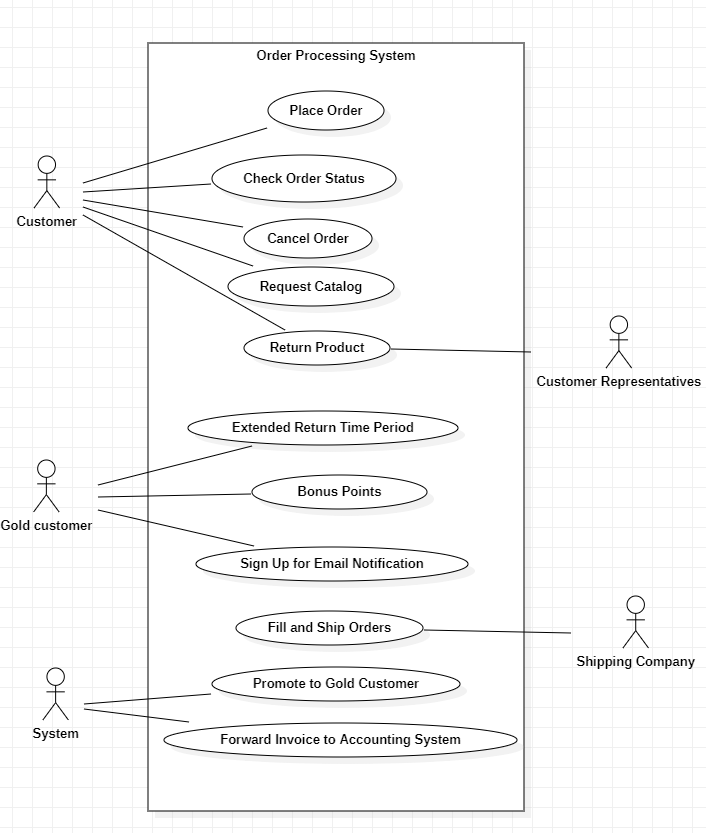
1. Place Order: Customers (both registered and non-registered)
2. Check Order Status: Customers
3. Cancel Order: Customers
4. Request Catalog: Customers
5. Return Product: Customers and Customer Representatives
6. Promote to Gold Customer: System (automatically)
7. Extended Return Time Period: Gold Customers
8. Bonus Points: Gold Customers
9. Sign Up for Email Notification: Gold Customers
10. Fill and Ship Orders: Shipping Company
11. Forward Invoice to Accounting System: System

Therefore, the use case diagram should have the following actors and their respective use cases linked to them:

* Customers: Place Order, Check Order Status, Cancel Order, Request Catalog, Return Product
* Customer Representatives: Return Product
* Shipping Company: Fill and Ship Orders
* System: Promote to Gold Customer, Forward Invoice to Accounting System
* Gold Customers: Extended Return Time Period, Bonus Points, Sign Up for Email Notification

1. the use case diagram 1.0

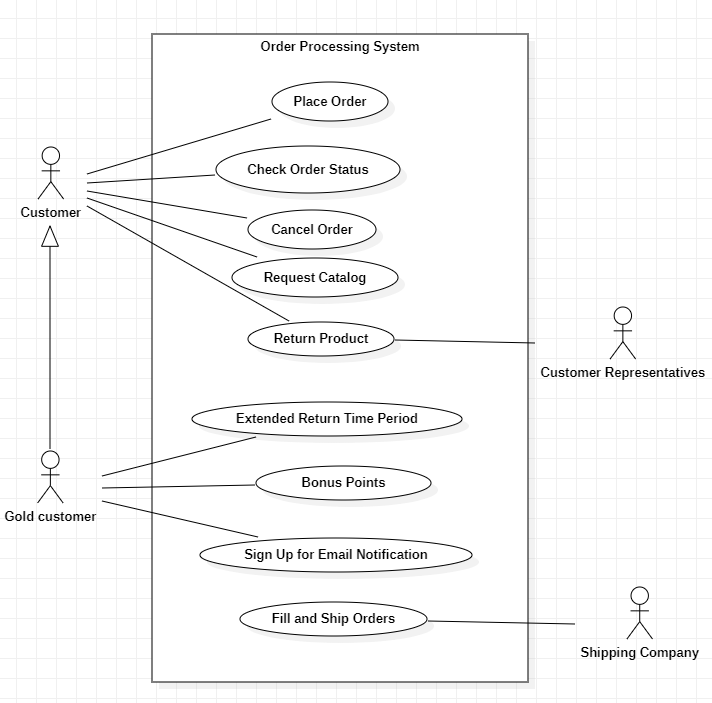
According to the answer of GPT, the diagram is as following :



1. Use case diagram 2.0

Because system shouldn’t be regard as actor, delete system actor and the associated use case.

The Gold customer generalize from Customer.



1. UML Class Diagram:

(1) chatGPT

Question 1:

Perform a quick application domain analysis to come up with an object model for the above system. Show the classes and critical attributes and methods

Here's an object model based on the system description provided:

Classes:

1. Customer
2. Product
3. Order
4. Catalog
5. CreditCard
6. ShippingAddress
7. BillingAddress
8. Inventory
9. Invoice
10. Account
11. BonusPoints

Critical Attributes and Methods:

1. Customer

* Attributes: customerId, name, email, phone, address, isGoldCustomer
* Methods: register(), getOrders(), getCatalog(), returnProduct()

1. Product

* Attributes: productId, name, description, price, image, isAvailable
* Methods: checkAvailability(), updateAvailability()

1. Order

* Attributes: orderId, customer, shippingAddress, billingAddress, products, totalCost, status
* Methods: addProduct(), removeProduct(), calculateTotalCost(), submitOrder(), cancelOrder(), checkOrderStatus()

1. Catalog

* Attributes: catalogId, products
* Methods: addProduct(), removeProduct(), updateProduct(), getProducts()

1. CreditCard

* Attributes: cardNumber, expirationDate, securityCode
* Methods: validate()

1. ShippingAddress

* Attributes: streetAddress, city, state, zipCode, country

1. BillingAddress

* Attributes: streetAddress, city, state, zipCode, country

1. Inventory

* Attributes: products, quantities
* Methods: updateQuantities()

1. Invoice

* Attributes: invoiceId, order, account, billingAddress, creditCard, totalCost
* Methods: generate()

1. Account

* Attributes: accountId, balance, bonusPoints
* Methods: updateBalance(), addBonusPoints()

1. BonusPoints

* Attributes: pointsPerDollar, pointsPerPurchase, expirationDate
* Methods: calculateBonusPoints()

Question 2:

show the data type of these attributes and return type and input type of methods

Here are the data types of the attributes and the input and return types of the methods for the object model I proposed earlier:

1.Customer

Attributes:

customerId: string

name: string

email: string

phone: string

address: ShippingAddress

isGoldCustomer: boolean

Methods:

register(): void

getOrders(): Order[]

getCatalog(): Catalog

returnProduct(order: Order, product: Product): void

2.Product

Attributes:

productId: string

name: string

description: string

price: number

image: string

isAvailable: boolean

Methods:

checkAvailability(): boolean

updateAvailability(isAvailable: boolean): void

3.Order

Attributes:

orderId: string

customer: Customer

shippingAddress: ShippingAddress

billingAddress: BillingAddress

products: {product: Product, quantity: number}[]

totalCost: number

status: string

Methods:

addProduct(product: Product, quantity: number): void

removeProduct(product: Product): void

calculateTotalCost(): number

submitOrder(): void

cancelOrder(): void

checkOrderStatus(): string

4.Catalog

Attributes:

catalogId: string

products: Product[]

Methods:

addProduct(product: Product): void

removeProduct(product: Product): void

updateProduct(product: Product): void

getProducts(): Product[]

5.CreditCard

Attributes:

cardNumber: string

expirationDate: string

securityCode: string

Methods:

validate(): boolean

6.ShippingAddress

Attributes:

streetAddress: string

city: string

state: string

zipCode: string

country: string

7.BillingAddress

Attributes:

streetAddress: string

city: string

state: string

zipCode: string

country: string

8.Inventory

Attributes:

products: Product[]

quantities: number[]

Methods:

updateQuantities(product: Product, quantity: number): void

9.Invoice

Attributes:

invoiceId: string

order: Order

account: Account

billingAddress: BillingAddress

creditCard: CreditCard

totalCost: number

Methods:

generate(): void

10.Account

Attributes:

accountId: string

balance: number

bonusPoints: BonusPoints

Methods:

updateBalance(amount: number): void

addBonusPoints(points: number): void

11.BonusPoints

Attributes:

pointsPerDollar: number

pointsPerPurchase: number

expirationDate: string

Methods:

calculateBonusPoints(totalCost: number): number

Question 3:

show the relation between classes

Here's a summary of the relationships between the classes in the object model:

Customer

* Has a ShippingAddress and BillingAddress
* Places Orders
* Has a Catalog

Product

* Is part of an Order
* Is part of a Catalog
* Has an Inventory quantity

Order

* Belongs to a Customer
* Has Products
* Has a ShippingAddress and BillingAddress
* Generates an Invoice upon submission
* Can be cancelled

Catalog

* Has Products

CreditCard

* Used for payment in an Invoice

ShippingAddress

* Used for shipping in an Order

BillingAddress

* Used for billing in an Order and Invoice

Inventory

* Tracks the quantity of Products

Invoice

* Belongs to an Account
* Has an Order
* Has a BillingAddress and CreditCard
* Generates a total cost

Account

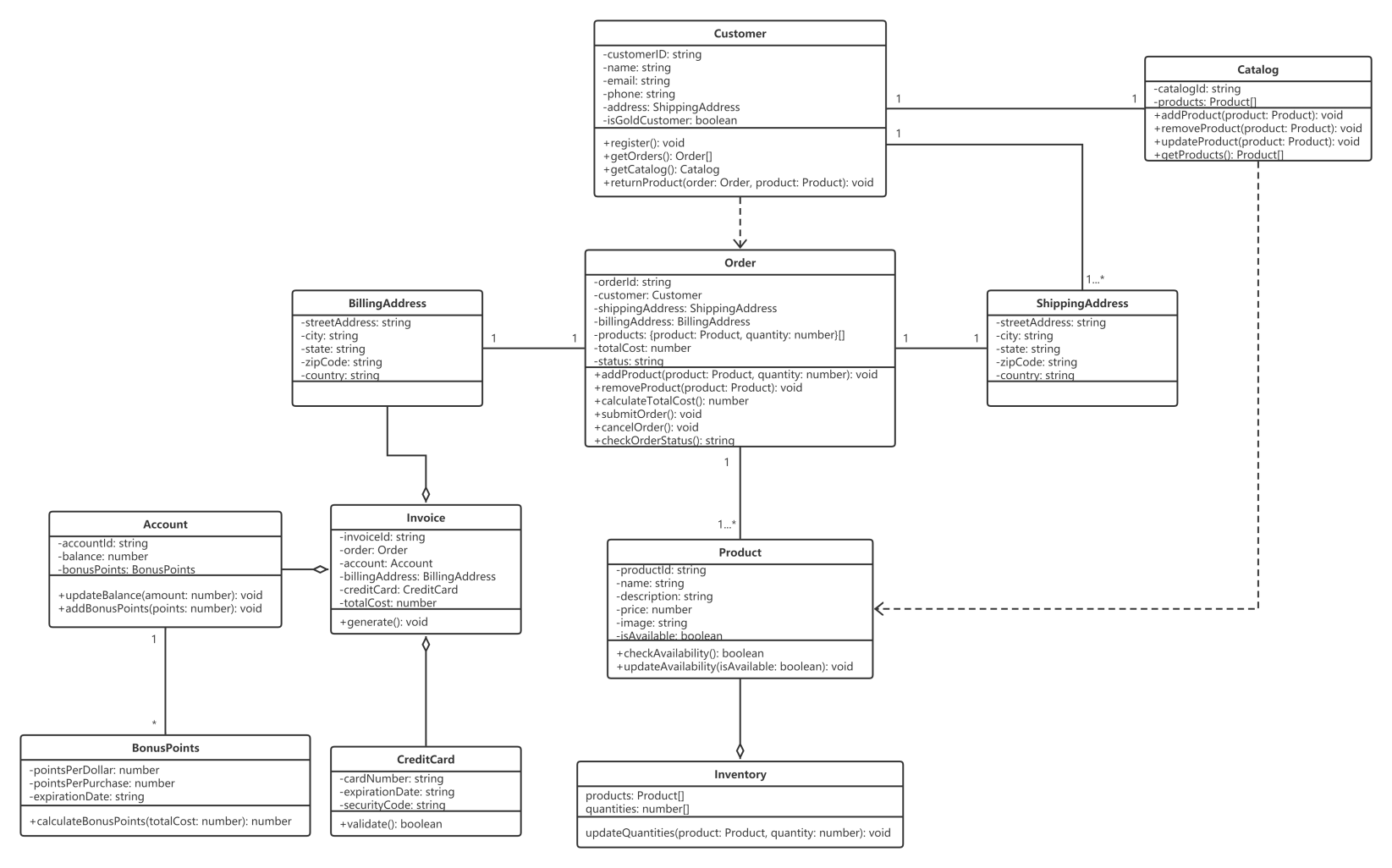
* Has an Invoice
* Tracks a balance and BonusPoints

BonusPoints

* Belongs to an Account
* Calculates points based on a total cost

1. The class diagram

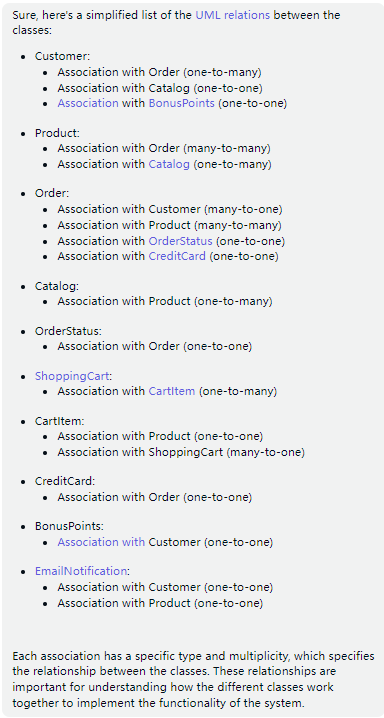
Draw the class diagram as following according to the description of chatGPT.



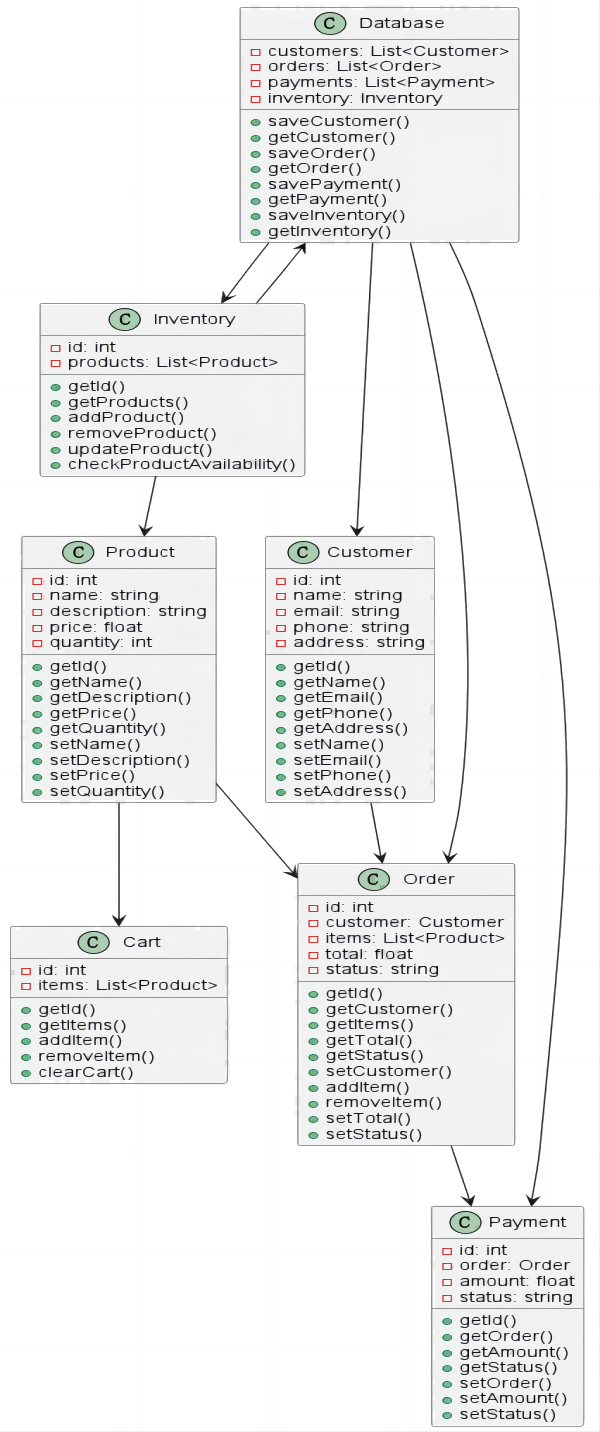
1. Try to ask gpt the uml relations between the classes:



The answer are all association. (and because I opened another conversation, the classes it gives to me are different.)



The class diagram drawn by plantuml of the changed classes:



And the code generalized by chatGPT:

@startuml

class Customer {

-id: int

-name: string

-email: string

-phone: string

-address: string

+getId()

+getName()

+getEmail()

+getPhone()

+getAddress()

+setName()

+setEmail()

+setPhone()

+setAddress()

}

class Product {

-id: int

-name: string

-description: string

-price: float

-quantity: int

+getId()

+getName()

+getDescription()

+getPrice()

+getQuantity()

+setName()

+setDescription()

+setPrice()

+setQuantity()

}

class Cart {

-id: int

-items: List<Product>

+getId()

+getItems()

+addItem()

+removeItem()

+clearCart()

}

class Order {

-id: int

-customer: Customer

-items: List<Product>

-total: float

-status: string

+getId()

+getCustomer()

+getItems()

+getTotal()

+getStatus()

+setCustomer()

+addItem()

+removeItem()

+setTotal()

+setStatus()

}

class Payment {

-id: int

-order: Order

-amount: float

-status: string

+getId()

+getOrder()

+getAmount()

+getStatus()

+setOrder()

+setAmount()

+setStatus()

}

class Inventory {

-id: int

-products: List<Product>

+getId()

+getProducts()

+addProduct()

+removeProduct()

+updateProduct()

+checkProductAvailability()

}

class Database {

-customers: List<Customer>

-orders: List<Order>

-payments: List<Payment>

-inventory: Inventory

+saveCustomer()

+getCustomer()

+saveOrder()

+getOrder()

+savePayment()

+getPayment()

+saveInventory()

+getInventory()

}

Customer --> Order

Product --> Cart

Product --> Order

Order --> Payment

Inventory --> Product

Inventory --> Database

Database --> Customer

Database --> Order

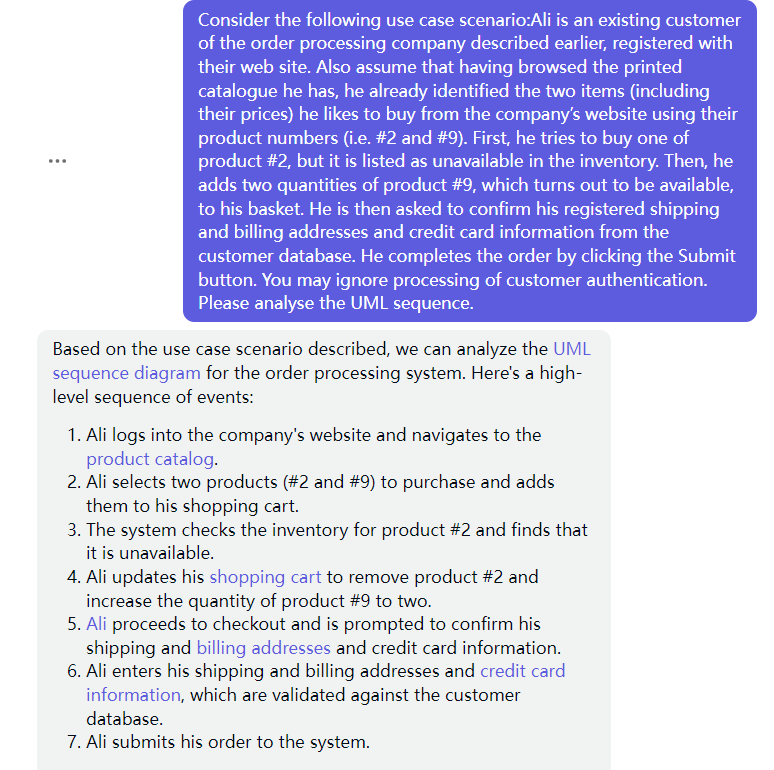
Database --> Payment

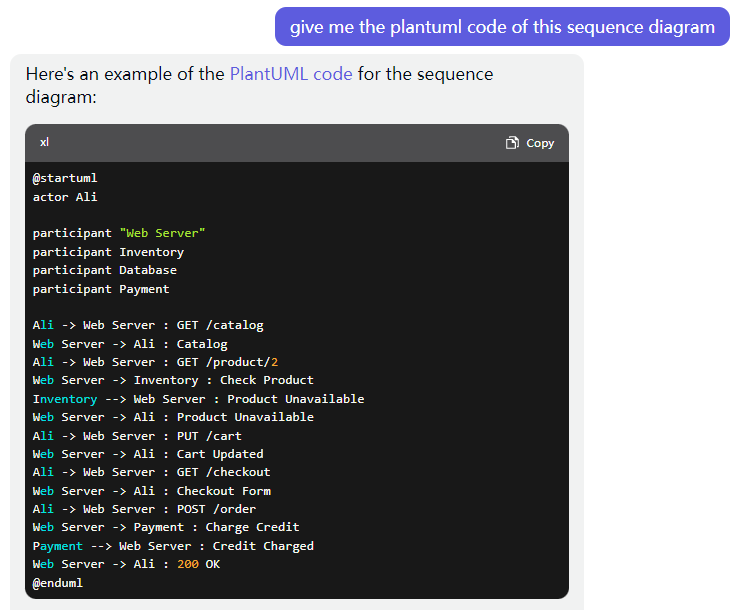
Database --> Inventory

@enduml

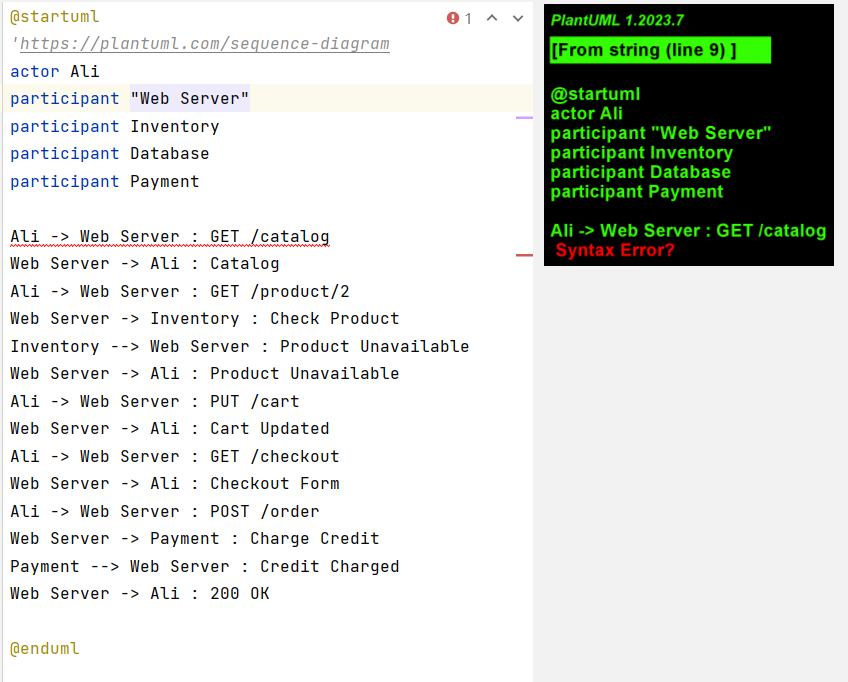
3. The sequence diagram:

1. chatGPT

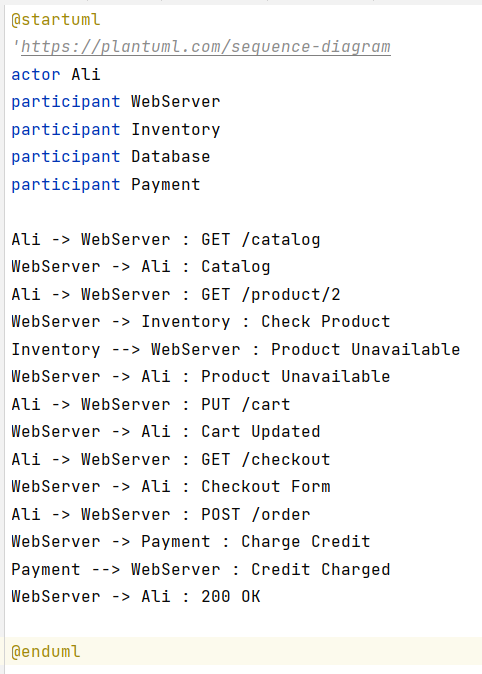




Try to draw the diagram using plantUML, but get error:



The code change to:



The diagram:

