Cairo University **Faculty of Computers and Artificial Intelligence**

Software design specification document

2023/2024

Project Team

ID	Name	Email
20211111	Hend Ahmed Kamal	ehend3343@gmail.com
20210117	Hazem Nasser Mohamed	hazemnasser3050@gmail.com
20210207	Abd-El Rahman Fawzy Sayed	Mamominy700@gmail.com

CS352: Sprint SDS–

Team: Postmen

Project: Orderly Notify

SDS document

Contents

Class diagram 3

Class diagram Explanation 4

Requirements Exposure as Web Service API 4

Requirement 5

Exposed API 5

Github repository link 12

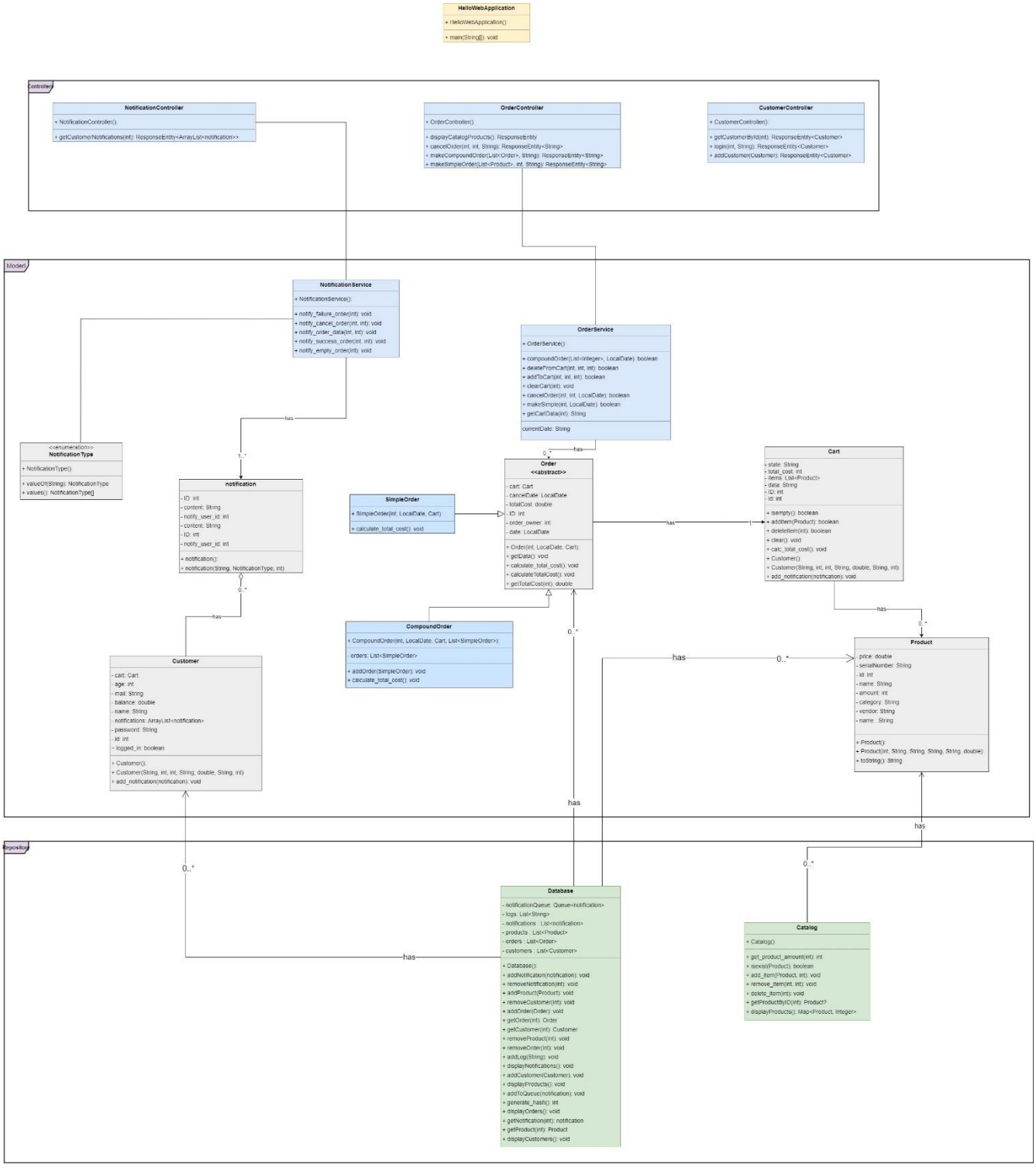
CS352: Sprint SDS–

Team: Postmen

Project: Orderly Notify

SDS document

Class diagram



CS352: Sprint SDS–

Team: Postmen

Project: Orderly Notify

SDS document

Class diagram Explanation

• Explain here the design pattern(s) that you used and your justification for using them, and the participating classes for each pattern.

Design Pattern	Classes	Explanation
1- Compound Design Pattern	Order Simple Order Compound Order	A compound order includes one or many simple orders.
2- Observer Design Pattern	Notification Notification Type Notification Service	One notification class is responsible for notifying all the customers

Requirements Exposure as Web Service API

Part 1: JSON file is attached with the zip file

SDS document

Part 2:

Explain here the exact mapping between every single requirement and its corresponding web service API operation. A sample example is provided to better explain the concept.

Requirement		Exposed API
Sign Up The system allows the customer to sign up and enter the following attributes it a JSON post request: 1- Name 2- Id 3- Email 4- Balance 5- Age 6- Password 7- Cart ID		1- http://localhost:8080/customers/add A service to allow the customer to sign up to the system. A post request that returns if the user signed up successfully to the system or not. Input: JSON object. Input example: `{ "name": "Walter White", "age": 30, "id": 55, "mail": "white@gmail.com", "balance": 500.0, "password": "exer52148", "cart": { "id": 55 } }`

SDS document

Output example: Customer type response entity


```
{
    "name": "Walter White",
    "password": "exer52148",
    "id": 2072114310,
    "notifications": [],
    "cart": {
        "id": -604521825,
        "total_cost": 0,
        "items": [],
        "state": "cart is empty !",
        "data": "cart is empty !these are cart data",
        "empty": true
    },
    "mail": "white@gmail.com",
    "balance": 500.0,
    "logged_in": false,
    "_data": "ID: 2072114310, Name: Walter White, Age: 30, Mail: white@gmail.com"
}
```


Log in.

The system allows the customer to log in with their id generated by the system and password.

1- http://localhost:8080/customers/login/{{customerID}}/{{password}}

A service to allow the customer to log in to the system. A Get request that returns the customer object if they logged in successfully.

Input: customer id and password in the URL
Input example: http://localhost:8080/customers/login/2072114310/exer52148

Output example:


```
{
    "name": "Walter White",
    "password": "exer52148",
    "id": 2072114310,
    "notifications": [],
    "cart": {
        "id": -604521825,
        "total_cost": 0,
        "items": [],
        "state": "cart is empty !",
        "data": "cart is empty !these are cart data",
```

SDS document


```
        "empty": true
    },
    "mail": "white@gmail.com",
    "balance": 500.0,
    "logged_in": true,
    "_data": "ID: 2072114310, Name: Walter White, Age: 30, Mail:
white@gmail.com"
}
```


View Catalog

The system allows the customer to view available products in the catalog and their quantity in the store, their vendor, type, price, name, and serial number.

1- http://localhost:8080/orders/products

A service to allow the customer to view the catalog of available products. A Get request that returns a list of available products and their details as a response entity.

Input: http://localhost:8080/orders/products

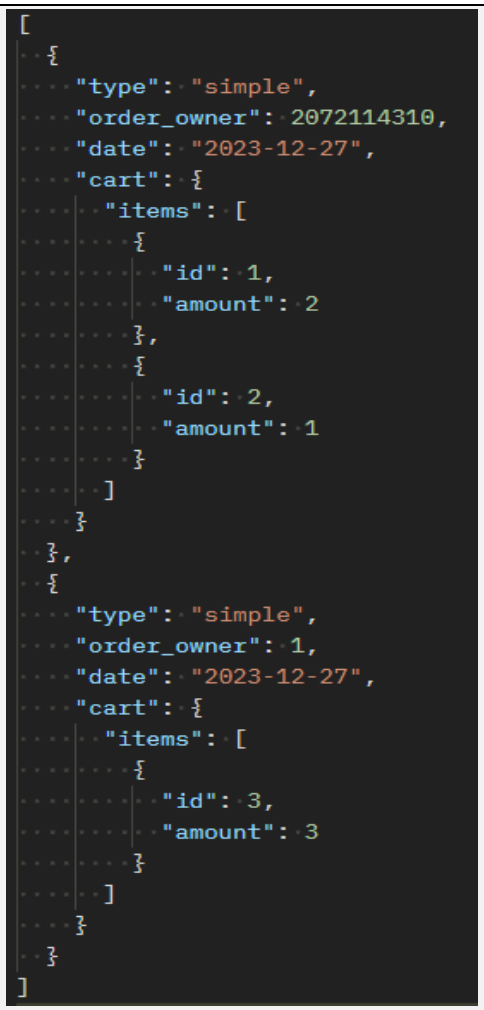
Output: a list of available products and their details


```
{
    "Product{id=6, name='Yogurt', serialNumber='131', vendor='Dairy', category='Yogurt', price=30.0}": 60,
    "Product{id=1, name='Bread', serialNumber='123', vendor='Bakery', category='Bread', price=5.0}": 10,
    "Product{id=4, name='Butter', serialNumber='101', vendor='Dairy', category='Butter', price=20.0}": 40,
    "Product{id=2, name='Milk', serialNumber='456', vendor='Dairy', category='Milk', price=10.0}": 20,
    "Product{id=5, name='Cheese', serialNumber='112', vendor='Dairy', category='Cheese', price=25.0}": 50,
    "Product{id=7, name='Apple', serialNumber='415', vendor='Fruits', category='Apple', price=35.0}": 70,
    "Product{id=3, name='Eggs', serialNumber='789', vendor='Dairy', category='Eggs', price=15.0}": 30
}
```

SDS document

Make simple order. The system allows the customer to make a simple order if they previously logged-in to the system and have account.		1- http://localhost:8080/orders/simpleOrder/{{customerID}}/{{placementDate}} A service to allow the customer to make a simple order. A Post request that returns a String as a response entity shows if the order is placed successfully or other exceptional cases like (user is not found within the system – product id is not found). Input: URL includes customer id and the current date of order placement Input example: http://localhost:8080/orders/simpleOrder/2072114310/2023-12-25 Output: String as a response entity shows if the order is placed successfully or other exceptional cases like (user is not found within the system – product id is not found). Output example: "Simple order placed successfully." Post conditions: • Customer balance is decreased. • Product amount is decreased. • Customer cart is filled with the products of the order. • A confirmation notification is sent to the customer with the order data and if the order is successfully being shipped or exceptional cases like if the cart is empty.
Make compound order. The system allows the customer to make a compound order if they previously logged-in to the system and have account. System checks if each customer in		1- http://localhost:8080/orders/compoundOrder/{{placementDate}} A service to allow the customer to make a compound order. A Post request that returns a String as a response entity shows if the order is placed successfully or other exceptional cases like (user is not found within the system – product id is not found). Input: URL includes customer id and the current date of order placement and a JSON object of more than one Order object each for one customer. Input URL: http://localhost:8080/orders/compoundOrder/2023-12-25 Input Body:

SDS document

the compound order is previously logged in the system, the order maker should have each customer id previously as an authentication.		 Post conditions: • Each Customer balance is decreased. • Each Product amount is decreased. • Each Customer cart is filled with the products of the order. • A confirmation notification is sent to each customer with the order data and if the order is successfully being shipped or exceptional cases like if the cart is empty.

SDS document

Show Customer Notifications. The system allows the customer to view their notification list that contains the information about order states, order id, order content, and remaining balance. System checks if the user is previously logged in.		1- http://localhost:8080/notifications/{{customerID}} A service to allow the customer to show their notifications. A Get request that returns a list of customer notifications as a response entity shows. Input: URL includes customer id Input URL example: http://localhost:8080/notifications/2072114310 Output: List of customer notifications Output example: [{ "notify_user_id": 2072114310, "content": "Dear: Walter White, your order ID is 71918163\norder data is: \nu have 2 items in the cartitem number : 1data is\nProduct{id=3, name='Eggs' serialNumber='789', vendor='Dairy', category='Eggs', price=15.0}item number : 1data is\nProduct{id=2, name='Milk', serialNumber='456', vendor='Dair category='Milk', price=10.0}these are cart data\nyour remaining balance : 475.0", "id": 101115062 }, { "notify_user_id": 2072114310, "content": "Dear: Walter White your order ID is 71918163Your order now is on SHIPPING state \nYou can cancel your order within 2 DAYS from now Current 2023-12-25", "id": -846945385 }, Post conditions: • Product amount is decreased if displayed.

SDS document

Cancel Order The system allows the customer cancel the simple or compound order within 2 days after the replacement date. **System checks if the user is previously logged in.**		1- http://localhost:8080/orders/cancelOrder/{{customerID}}/{{orderID}}/{{cancelDate}} A service to allow the customer to cancel their orders. Delete request that returns a String if the order is cancelled successfully or the customer exceeded the cancellation period. Input: URL includes customer id, order id, and cancellation date. Input URL example: http://localhost:8080/orders/cancelOrder/2072114310/71918163/2023-12-30 Output: String shows if the order is cancelled successfully or the customer exceeded the cancellation period. Output example: Bad scenario: “You have exceeded the cancellation period.” Happy scenario: “Order Cancelled successfully” Post conditions: • Order total cost is returned to the customer balance. • Each product amount of the order is returned to the store. • A cancellation notification is sent to the customer.

SDS document

Notification queue
The system allows the notification queue of the database to pop a notification each 40 second to simulate it is sent to the user.
		This requirement is implemented using a scheduling algorithm.
		`// Create a scheduled executor service`
		`ScheduledExecutorService executorService = Executors.newScheduledThreadPool( corePoolSize: 1);`
		
		`// Schedule a task to run every 5 seconds`
		`executorService.scheduleAtFixedRate(() -> {`
		`    // Check if the notification queue is not empty`
		`    if (!Database.getNotificationQueue().isEmpty()) {`
		`        // Pop the first notification from the queue`
		`        Database.getNotificationQueue().poll();`
		`    }`
		`}, initialDelay: 0, period: 40, TimeUnit.SECONDS);`

Github repository link

https://github.com/AbdEl-Rahman-Fawzy/Ordering_and_Notify.git