Cairo University

Faculty of Computers and Artificial Intelligence

Assignment 2

Software design specification document

2023/2024

Project Team

ID	Name	Email
20210147	ZIAD AYMAN FAROUK	zeromaradona@gmail.com
20210042	AHMED NAGY RAMADAN	anagy464@gmail.com
20200679	OMAR EHAB MOHAMED EL-HAREDY	omarelharedy12@gmail.com

CS352: Sprint SDS– Task Force X
SDS document

Contents

Class diagram design **2**

Class diagram Explanation **3**

Requirements Exposure as Web Service API **4**

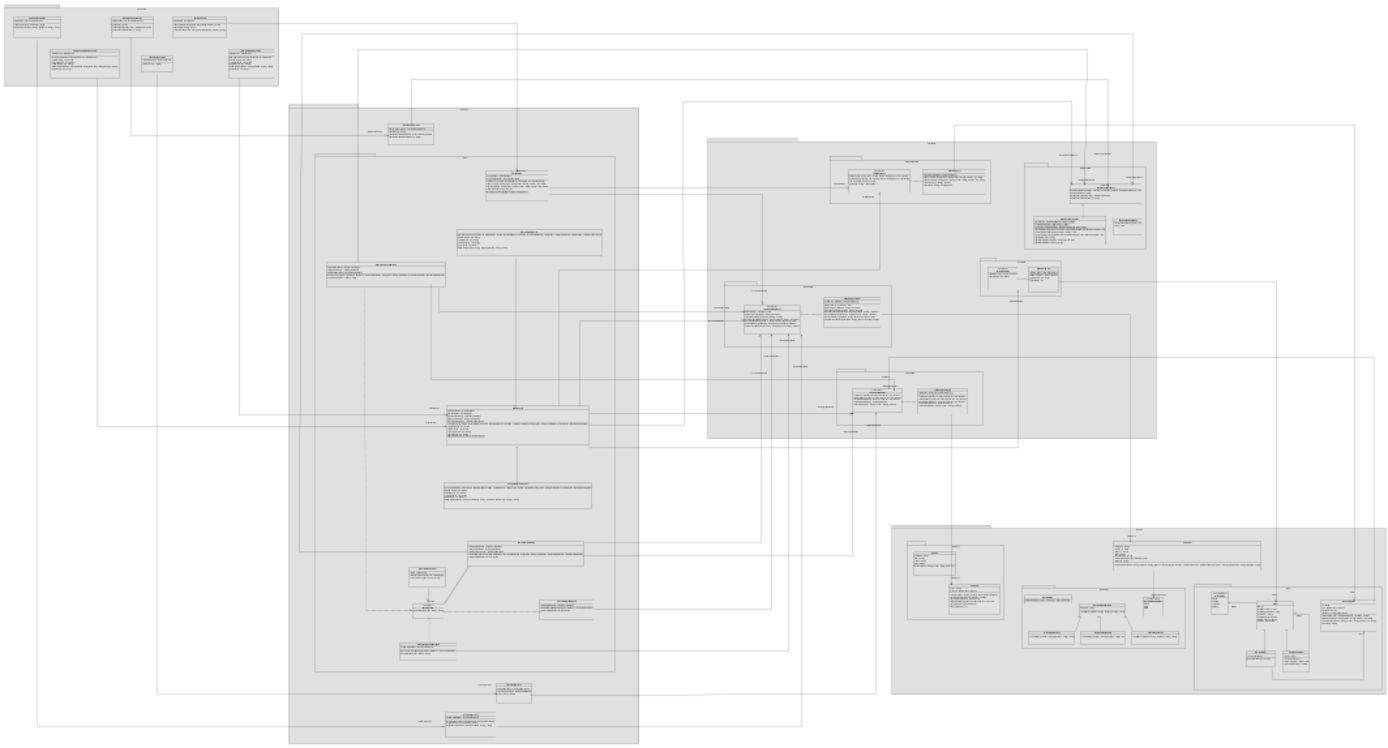
- Exposed Postman Collection **4**

  A. Customer Requests 4
  B. Cart Requests 4
  C. Inventory Requests 5
  D. Compound Order Requests 5
  E. Notification 5
  F. Simple Order Request **5**

Class diagram design

CS352: Sprint SDS– **Task Force X**

SDS document



There is a PNG file in the zip file with better quality.

File Name: ClassDiagram_20210042_20210147_20200679.png

Class diagram Explanation

1) Composite pattern in order

- **Participants in the Composite Pattern:**
  - **Component (Order):**
    - Declares the interface for objects in the composition.
    - Implements default behavior for the interface common to all classes (e.g., getTotalProdPrice).
    - Declares an interface for accessing and managing child components.
  - **Leaf (SimpleOrder):**
    - Represents leaf objects in the composition.
    - Implements the behavior for primitive objects.
  - **Composite (CompoundOrder):**
    - Defines behavior for components that have children.
    - Stores child components.
    - Implements the child-related operations in the Component interface.
- **Reason for using composite pattern:**
  The pattern is suitable for representing hierarchical structures, like orders and suborders, where a composite order can contain multiple simple orders.

2) Strategy in order cancellation

- **Participants in the strategy Pattern**
  - **OrderActionContext (Context):**
    - Represents the context that uses the strategy.
    - Contains a reference to an IOrderAction strategy.
    - Executes the strategy by calling the performAction method.
  - **IOrderAction (Strategy Interface):**
    - Declares the interface for the strategy with the performAction method.
  - **Concrete Strategies:**
    - CancelSimpleShipped
      - Implements the strategy for canceling a simple order that has been shipped.
    - CancelSimplePlaced
      - Implements the strategy for canceling a simple order that is placed.
    - CancelCompoundShipped

        - Implements the strategy for canceling a compound order that has been shipped.
    - CancelCompoundPlaced
        - Implements the strategy for canceling a compound order that is placed.

➢ **Reason for using Strategy pattern:**
The Strategy pattern provides a way to define a family of algorithms, encapsulate each one, and make them interchangeable.

3) Strategy in Notification Creation

➢ **Participants in the strategy Pattern**

- **InMemoryNotification (Context)**
    - Represents the context that uses the strategy.
    - Executes the strategy by calling the createBody method.
- **NotificationTemplate (Strategy Interface):**
    - Declares the interface for the strategy with the createBody method.
- **Concrete Strategies:**
    - PlacedNotification
        - Implements the strategy for canceling a simple order that has been shipped.
    - CancelNotification
        - Implements the strategy for canceling a simple order that is placed.
    - ShippedNotification
        - Implements the strategy for canceling a compound order that has been shipped.
    - CancelCompoundPlaced
        - Implements the strategy for canceling a compound order that is placed.

➢ **Reason for using Strategy pattern:**
The Strategy pattern provides a way to define a family of algorithms, encapsulate each one, and make them interchangeable.

Requirements Exposure as Web Service API

Exposed Postman Collection

Customer Requests

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31763526-f34d95e1-9b0e-4953-bdf4-a406f68b108d?action=share&creator=31763526

Cart Requests

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31763526-bcfda7ed-586d-4edd-bb9a-755eaba6b4fe?action=share&creator=31763526

Inventory Requests

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31763526-e9609217-c081-42d9-a3f8-274beba4752d?action=share&creator=31763526

Compound Order Requests

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31763526-3befa34c-01ff-4037-9492-18171a168960?action=share&creator=31763526

Simple Order Request

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31763526-34e20671-a586-451e-9f78-01f1e34ba2c8?action=share&creator=31763526

Notification Requests

Postman Collection Link: https://www.postman.com/task-force-2456/workspace/software2-api-assignment/collection/31928807-cb14c366-87f7-42fc-8aac-51df654c722d?action=share&creator=31763526

Postman workspace link (in case one of the above links didn't work) : https://www.postman.com/task-force-2456/workspace/software2-api-assignment

Requirement		Exposed API
➔ **A customer should be able to create an account and put a specific balance using that account.** ➔ **Such balance would be utilised during future purchasing operations.**		**POST/Register** This API endpoint is used to register a user. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the raw request body: ● username : The username of the user to be registered. ● password : The password for the user account. ● balance : The initial balance for the user account. ● NotificationChannel:email or sms or both ● email ● phoneNumber ● language
➔ **A customer should be able to log into an account.**		**GET/login/{username}/{password}** This API endpoint is used to login a user. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the URL: ● username : The username of the user to be logged in. ● password : The password for the user account.

• **The "Orders and Notifications Management" module should provide the following features:** ○ A list of all the products currently available for purchase should be displayed. ○ Products are defined by a serial number, a name, a vendor, a category, and a price. Furthermore, a count of the remaining parts from each category should be available.		**GET/get-inventory** This endpoint request to retrieve the inventory. The request does not include a request body. **Response** The response returns a status code of 200, along with an array containing inventory information. Each item in the array includes the total number of parts, name, and a list of products with details such as vendor, price, parts number, and product name.
➔ **A customer can place a simple order, where such an order would include a single product or several products..You should be able to list all the details of simple order** ➔ **Once the order is placed, the balance of each order is deducted from its corresponding customer.** ➔ **After placing the order, the user can ship the order. For simple orders, when an order is shipped,ts shipping fees are deducted from its customer's account.**		**POST/add-to-cart/{username}** This API endpoint is used to add a product to cart. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the raw request body: • name : name of the product • vendor: vendor of the product • amount : how many items of the product **GET/get-cart/{username}** This endpoint makes a GET request to retrieve the cart details for a specific user. The response to the last execution of this request had a status code of 200, and it returned the cart details in a map containing total price, id and a map of products.

		POST/simple-order/add/{username}/{address} This API endpoint is used to add an order to the orders database. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the URL: ● username : The username of the user. ● address : address to be shipped to. **POST/simple-order/place/{orderID}** This API endpoint is used to check the balance of the user and the stock of the items in the order if both are sufficient the total cost of the items is deducted from the user balance , the number of the items in the orders is deducted from the inventory and the order status is changed to placed meaning ready for shipment then add order to the orders database. Upon successful execution it returns a status code of 200 with a String response. The request should include the following parameters in the URL: ● orderID **POST/simple-order/ship/{OrderID}** This API endpoint is used to check the balance of the user and the status of the order. If both are sufficient the cost of the shipment is deducted from the user balance and the order status is changed to shipped . Upon successful execution, it returns a status code of 200 with a String response.

		The request should include the following parameters in the URL: ● orderID
➔ **a customer can make a compound order, where that order can include various orders headed to near-by locations, in addition to his own products, to lessen the shipping fees. For instance, a customer can make a compound order for himself and two of his friends, who all live in different locations in El-Dokki** ➔ **You should be able to list all the details of compound orders.** ➔ **To place a compound order, the customer needs to pick his products, and his friends' orders as well. Once the order is placed, the balance of each order is deducted from its corresponding customer.** ➔ **After placing the order, the user can ship the order.For compound orders, when an order is shipped, its shipping fees are deducted from all the customers who participated in the compound order.**		**GET/compound-order/get/{OrderID}** This API endpoint makes a GET request to retrieve details of a specific compound order identified by the ID parameter in the URL. The response will be an array of orders including information about the total product price, shipping fees, order ID, shipping date, items in the cart (including quantity, vendor, price, and name), customer details, order status, and shipping addresses. **POST/compound-order/add/{username}/{address}** This API endpoint is used to add a CompoundOrder to the orders database Upon checking the users existence. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the RequestBody: ● username : The username of the user. ● address : address to be shipped to. ● Customers map: username key and an address value **POST/compound-order/place/{OrderID}** This API endpoint is used to check the balance of the users and the stock of the items in the order and the order status if all is sufficient the cost of each order is

		deducted from its user balance , the number of the items in the orders is deducted from the inventory and the order status is changed to placed meaning ready for shipment. Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the URL: ● orderID **POST/compound-order/ship/{OrderID}** This API endpoint is used to check the balance of the user and the status of the order. If both are sufficient the cost of the shipment is deducted from the user balance and the order status is changed to shipped . Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the URL: ● orderID
➔ **Notification creation for various operations. The system manages those different notification"templates" and the languages these templates can be sent in, and the "placeholders"within the content of these templates to be replaced with "actual values".**		**the creation of the template messages happens and is handled within the placeOrder and ShipOrder APIs ; after everything is confirmed the proper notification template is created and placeholders are replaced with actual values**

➔ **For created notifications, you should implement a "notifications Queue", where you insert** • **"notifications" that ARE TO BE SENT. You should be able to list the current content of that Queue.**		**GET/notifications/queue** This API endpoint is used to view the message in the queue. Upon successful execution, it returns a status code of 200 with a String response. **The request doesn't include parameters in the URL nor the body.**
Bonus ➔ **Customers can cancel an order placement, or cancel only its shipping within a pre-configured automated duration.** ➔ **Such cancellation should update appropriately within the system.** ➔ **After a configured time, messages are removed from the queue automatically to simulate that they were actually sent. Accordingly, the system should provide some live statistics to the overall software.** ➔ **The target of these statistics is to provide a vision about the notifications that are sent successfully:** **• The most notified email-address/phone-number.** **• The most sent notification template.**		**GET/simple-order/cancel/{OrderID}** This API endpoint is used to cancel an order, it checks the status of the order if placed it simply just increases the balance of the customer back , increases the stock back and changes the status to cancelled. If shipped checks within the permissible duration for cancellation if sufficient it changes the status back to placed if not it refuses the operation Upon successful execution, it returns a status code of 200 with a String response. The request should include the following parameters in the URL: • orderID **GET/notifications/MostNotifiedCustomer** This API endpoint is used to get the most Notified customer i.e. email address or phone number. It takes no input.

		GET/notifications/MostNotifiedTemplate This API endpoint is used to get the most used notification template for example: the placed notification template or the shipped notification template. It takes no input.