Assignment- 2
(API Exercise)

Due Date: October 30th, 2022 and upload your files to your designated OneDrive folder. To be accomplished individually

SCOPE:

The following exercise is for you to demonstrate your ability to write a small micro-service from an API specification. Below is the summary of the tasks required to complete this exercise. Please note that the scenario below is fictional and does not reflect the API of any real-world business.

1) **Context**
   The task will include setting up a new project and writing an API implementation to the specification below. It is recommended that you use Spring boot and NodeJS as the technologies for the new API.

2) **Problem Description**
   A large e-commence company is working on new back-end APIs for their shopping cart.

- **Product MicroService:** They have designed 1 new API that need to be built in as a micro-service,
    a. **retrieve list of Products API,** this api will return list of products from DB
        i. Product Object Attributes (with Example):
            1. "productId": "12445dsd234",
            2. "category": "Modile",
            3. "productName": "Samsung",
            4. "productModel": "GalaxyNote",
            5. "price": 700
            6. "availableQuantity": 10

  For simplicity do not create any validation/Test on this Microservice.

- **User Cart MicroService:** They have designed 2 new APIs that need to be built in this micro-service,
    a. **Add/Update items to cart API:** a PUT call to add or update items and its quantity for selected products into DB.
        i. CartItem Object Attributes (with Example)
            1. "productId": "12445dsd234",
            2. "productName": "Samsung",
            3. "quantity": 2
            4. "amount": 1400,

ii. Basic Flow
    1. On PUT request.
    2. Validate user, consider few hard coded users for now.
    3. Call Product MicroService to get All Products
    4. Validate if sufficient quantity is available of product
    5. Get Existing CartItem for this product
    6. If CartItem exists, update with addition quantity, update DB
    7. Otherwise create new Cart Item. Add to DB
    8. Send added or updated CartItem to client.

b. **Retrieve items in cart:** a GET call to retrieve items and its quantity for user
    i. CartItem Object Attributes: same as above

    ii. Basic Flow
        1. On GET request.
        2. Validate user, consider few hard coded users for now.
        3. Retrieve all CartItem from DB for given user.
        4. Send response back to client

3) Setup

- Create a Git repository. Check-in your code and send us your repo URL.
- Create separate projects for both Microservices.
- Write a README.md with all the instructions to install, test and run your code.

4) Relaxation

- You can use Java or NodeJs or any coding language
- Can use Java collection Framework or create in-Memory Data Structure for Node JS to store data (work as in-Memory DB), you are free to build your own data model. Do not spend time to build Repositories.
- **Product MicroService** will act as downstream API for **User Cart MicroService, no validation required for Product MicroService**
- Write Test and validation only for **User Cart MicroService**.
- Avoid creating to many application level layers, few expected Layers: Controller, Service and Model.

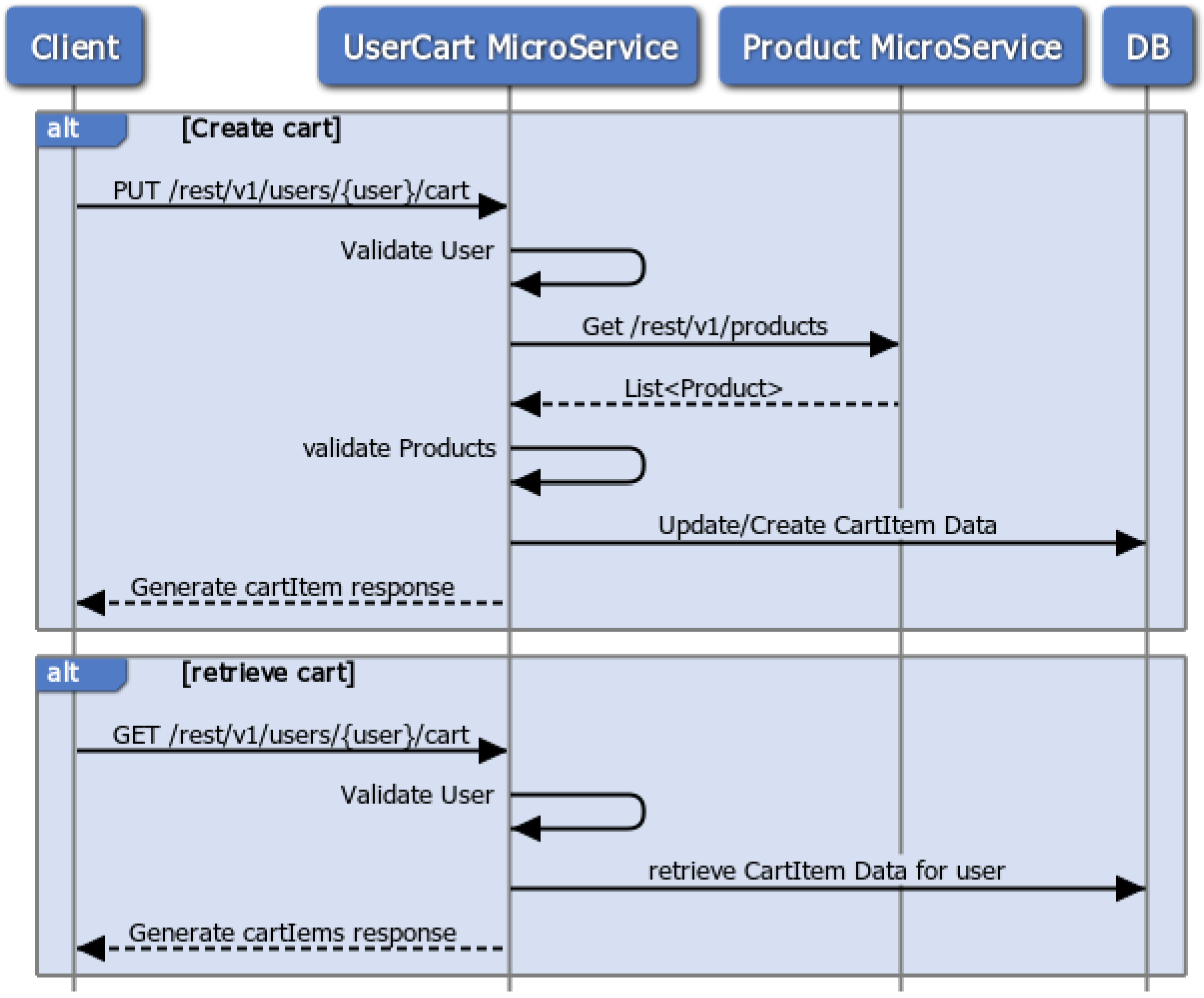
5) Estimated Exercise Duration

It is expected that you would need somewhere between 5-10 hours to complete this exercise. Feel free to use TODO comments (e.g. // TODO) where there might be scenarios or features that could be implemented in the future.

6) Version Control and Submission

Before starting the exercise, please create a new free Git account.
This will be the source control for your project, which you can commit into and push changes to the server as required.

7) Sequence Flow


Client
UserCart MicroService
Product MicroService
DB
alt
[Create cart]
PUT /rest/v1/users/{user}/cart
Validate User
Get /rest/v1/products
List<Product>
validate Products
Update/Create CartItem Data
Generate cartItem response
alt
[retrieve cart]
GET /rest/v1/users/{user}/cart
Validate User
retrieve CartItem Data for user
Generate cartIems response

8) Non-functional Considerations

Consideration should be given to the following areas:

- Configuration - How will you store configuration that might vary in different environments, for example the endpoint for the Get Product API. Are there frameworks that make this easy for us out of the box?
- How often would we call the Get Product API? Would we hit it every time, or would we think about caching the response for a period of time? *NB: We don't expect you to build a caching solution, simply think about it and we can discuss afterwards.*
- Security - How could we secure this type of API? Once again, building security is not part of this task.

9) New API Documentation

a. **GET Product API**: This is the API to be exposed in the Product micro-service. The service will return a list of porducts that are applicable to a given user. This API will act as downstream API to UserCart Microservice.

API Request
HTTP Method: GET
Request URI: /rest/v1/products

Query Parameters: NA
API Response
Response Schema: Responses will be application/json. Upon success, a list of Products objects will be returned. If an error occurs, an ErrorResource object will be returned instead.

Example Response:
This is the API we are asking you to build.
Request path variables are highlighted in RED below.

```
[
{
"productId": "12445dsd234",
"category": "Modile",
"productName": "Samsung",
"productModel": "GalaxyNote",
"price": 700
"availableQuantity": 10
},
{
"productId": "123245ds4234",
"category": "TV",
"productName": "Sony",
"productModel": "Bravia",
"price": 1200
"availableQuantity": 6
}
]
```

Fill free to user data type for above attributes

b. **PUT CartItem API:** This API will be used to create/update items to user cart with the products and its quantity. This is the API to be exposed in the UserCart micro-service

API Request
HTTP Method: PUT
Request URI: /rest/v1/users/<uuid>/cart
Body Parameter

```
{
"productId": "12445dsd234",
"quantity": 2,
}
```

Note: only one product can be added into cart at a time.

API Response
Response Schema: Responses will be application/json. Upon success, a CartResource object will be returned. If an error occurs, an ErrorResource object will be returned instead.

Example Request:
PUT /rest/v1/users/qa-test-user/cart
Example Response:
Request path variables are highlighted in RED below.

```
{
"productId": "12445dsd234",
"productName": "Samsung",
"quantity": 2
"amount": 1400,
}
```

c. **Retrieve UserCart API:** This will return all available products in the cart for given user. This is the API to be exposed in the UserCart micro-service

API Request
HTTP Method: GET
Request URI: /rest/v1/users/<uuid>/cart
API Response
Response Schema: Responses will be application/json. Upon success, a CartResource object will be returned. If an error occurs, an ErrorResource object will be returned instead.

Example Request:
GET /rest/v1/users/qa-test-user/cart
Example Response:
Request path variables are highlighted in RED below.

```
{
```

```
"uuid": "qa-test-user",
"cart": [{
"productId": "12445dsd234",
"productName": "Samsung",
"quantity": 2
"amount": 1400,
}
]
}
```

References:

- Coding Microservices in NodeJS https://nodesource.com/blog/microservices-in-nodejs
- Converting your APIs to REST standards https://www.codementor.io/@olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd
- https://www.twilio.com/blog/building-javascript-microservices-node-js
- To try out simple APIs without actually installing lot of tools. Click “try it out” one opening the website https://raml.org/
- Setting up NodeJS for WSL (ubuntu in windows 10) https://docs.microsoft.com/en-us/windows/nodejs/setup-on-wsl2