Interfaces

Since we are using the HTTP REST protocol, our interfaces are exposed as the APIs.

Catalog Microservice:

1. GET /catalog/<stock_name> (this API is exposed to frontend service)

   Description: API to lookup stock information by passing stock_name in the URL path

   Inputs: stock_name (received as a parameter in the URL path)

   Outputs: returning multiple responses (success and error) in JSON format.

   1) Output for success request
      GET /catalog/GameStart

```
{
    "name": "GameStart",
    "price": 100,
    "quantity": 54680
}
```

   2) Output if stock name not found
      GET /catalog/sample

```
{
    "error": "Stock Not Found!"
}
```

2. PUT /catalog (this API is exposed only to order microservice)

   Description: API to check if a stock is present or not and if present update(increment/decrement) the stock quantity based on the transaction type and increase the trading volume of the stock

   Inputs: accepts a JSON payload in the request body

   Sample JSON input:

```
{
    "name":"GameStart",
    "quantity": 20,
    "type": "buy"
}
```

Outputs: returns multiple responses (success or error) in JSON format

Sample output:

1. Output: For success request

   For sample input, {"name":"GameStart", "quantity": 20, "type": "buy"},

```
{
    "name": "GameStart",
    "price": 100,
    "quantity": 80,
    "trading_volume": 20
}
```

2. Output: For invalid stock name
   For sample input, {"name":"sample", "quantity": 20, "type": "buy"},

```
{
    "error": "Stock Not Found!"
}
```

3. Output: If the requested stock quantity exceeds the available stock quantity
   For sample input, {"name":"GameStart", "quantity": 50000, "type": "buy"},

```
{
    "error": "Quantity Exceeded Available Quantity!"
}
```

Order Microservice:

1. POST /orders (this API is exposed to frontend service)
   Description: API to trade stocks by sending stock name, quantity and transaction type. It then calls the catalog microservice to check if the stock is found and requested

quantity is available for buy type and updates (increments/decrements) the quantity accordingly in the catalog database and returns the updated stock details to the order microservice. The order microservice then logs the received response into its database and generates the unique transaction number and returns the transaction number to frontend in JSON format.

Input: Accepts inputs in request body in JSON format.

Sample JSON input:

```
{
    "name":"GameStart",
    "quantity": 20,
    "type": "buy"
}
```

Output: returns multiple responses(success or error) in JSON format

Sample output:

1. Output: For success request
   For sample input, {"name":"GameStart", "quantity": 20, "type": "buy"},

```
{
    "transaction_number": 1
}
```

2. Output: For invalid stock name
   For sample input, {"name":"sample", "quantity": 20, "type": "buy"},

```
{
    "error": "Stock Not Found!"
}
```

3. Output: If requested stock quantity exceeds the available stock quantity
   For sample input, {"name":"GameStart", "quantity": 50000, "type": "buy"},

```
{
    "error": "Quantity Exceeded Available Quantity!"
}
```