Simple Online Marketplace API

Implementation

This API allows seller add product, restock product and remove product from the online marketplace, customer can select products and put them in a shopping cart and buy them. Customer can also clear the shopping cart. Both seller and customer can view current products storage.

This small APP is written in Node.js with Express.js framework (for building the API), Jest and Supertest framework (for Unit test). The online marketplace storage information is saved in a local ***storage.json*** file and the shopping cart information is saved in a local ***cart.json*** file.

This app covers basic error checking and error handling which includes logical check and API check. Wrong actions should have proper error messages displayed to the user and invalid API request would result in 404 Bad Request Status with error message well printed.

The main file ***app.js*** does not have any logical section, instead, it calls corresponding functions from different ***js*** files. This way, we can achieve high cohesion, low coupling and more structured architecture.

I also wrote some basic unit tests which cover most of the functions and APIs.

Usage

Installation

Download and install the latest Node.js from https://nodejs.org/en/download/ .

Run

Once node.js is installed, navigate to ***online-marketplace-api*** folder and open the terminal, type in

npm install
npm start (or node app.js)

Once you see

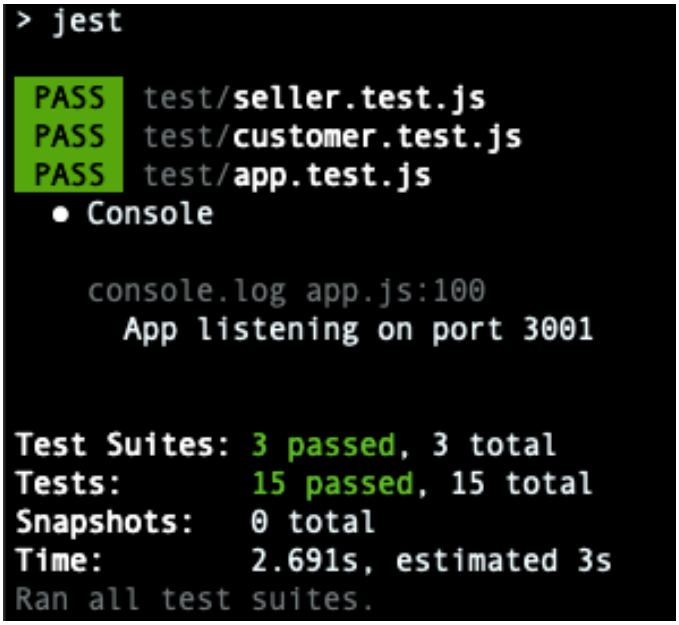
App listening on port 3001

the app is up and running.

Unit Test

To run unit test, do

npm test



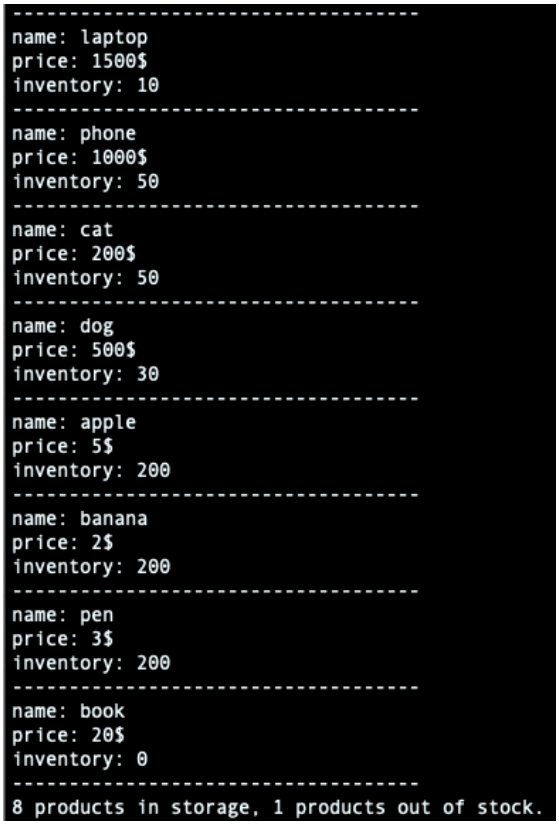
User story

1. General

These commands do not require specific role

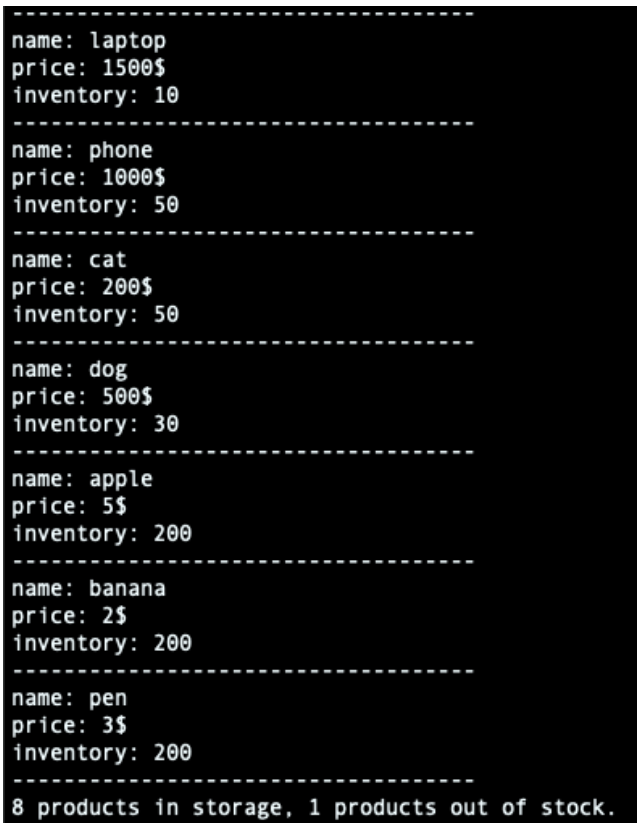
a. List all the products in the storage

curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/list'



b. List all the products in the storage that are available

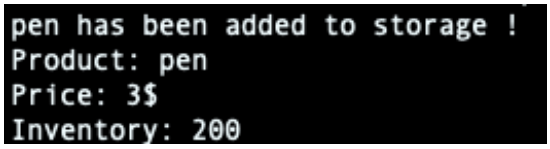
curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/list?filter=available'



2. Seller

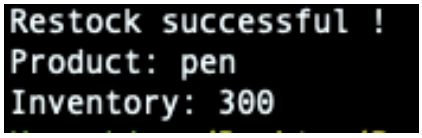
a. Seller add a product with product name, price and quantity.

curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/seller?action=add&name=pen&quantity=200&price=3'



b. Seller restock a product with name and quantity

curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/seller?action=restock&name=pen&quantity=100'



c. Seller remove a product with name

curl -X GET -H "Content-Type: application/json"
'http://localhost:3001/seller?action=remove&name=pen'

```
pen removed from storage !
```

3. Customer

a. Customer select items and put in shopping cart

curl -X GET -H "Content-Type: application/json"
'http://localhost:3001/customer?action=select&name=apple&quantity=10'

```
10 apple(s) has/haven been put in to shopping cart
```

b. Customer view current shopping cart

curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/customer?action=cart'

```
------------------------------------
name: apple
price: 5$
quantity: 10
------------------------------------
name: laptop
price: 1500$
quantity: 1
```

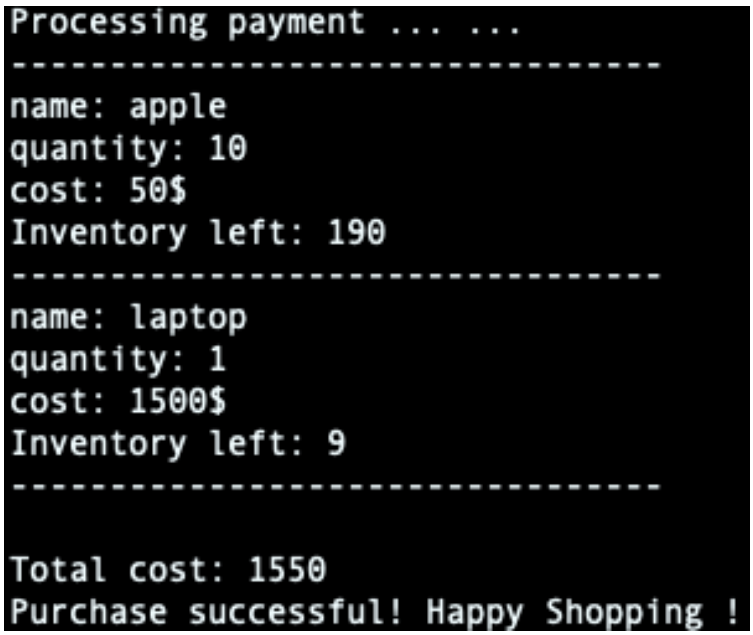
b. Customer clear the current shopping cart

curl -X GET -H "Content-Type: application/json"
'http://localhost:3001/customer?action=clear'

```
Shopping cart has been cleared!
```

b. Customer buy items in the current shopping cart

curl -X GET -H "Content-Type: application/json" 'http://localhost:3001/customer?action=buy'



Error Handling

This app covers following error cases and all of them are being tested in the unit tests. For details, please check the ***test*** folder.

Seller

1. Seller add product that is already exist

```
Product already exists!
```

2. Seller add product without specifying name

```
Error: Bad request
```

3. Seller add product without specifying price

```
Price missing!
```

4. Seller restock product that does not exist

```
car not found in the storage!
```

5. Seller remove product that does not exist

```
car not found in the storage!
```

6. Seller perform invalid action (eg, buy)

```
Action not allowed!
```

Customer

1. Customer select a product that does not exist

```
car not found in the storage!
```

2. Customer select a product that has been sold out

```
Sorry, book has been sold out!
```

3. Customer select a product with quantity that exceeds the inventory count

```
laptop only has 9 left !
```

4. Customer buy products when shopping cart is empty

```
Shopping cart is empty!
```

5. Customer perform invalid action (eg, add)

```
Action not allowed!
```

General

Invalid roles and invalid queries should bring Bad Request Error.