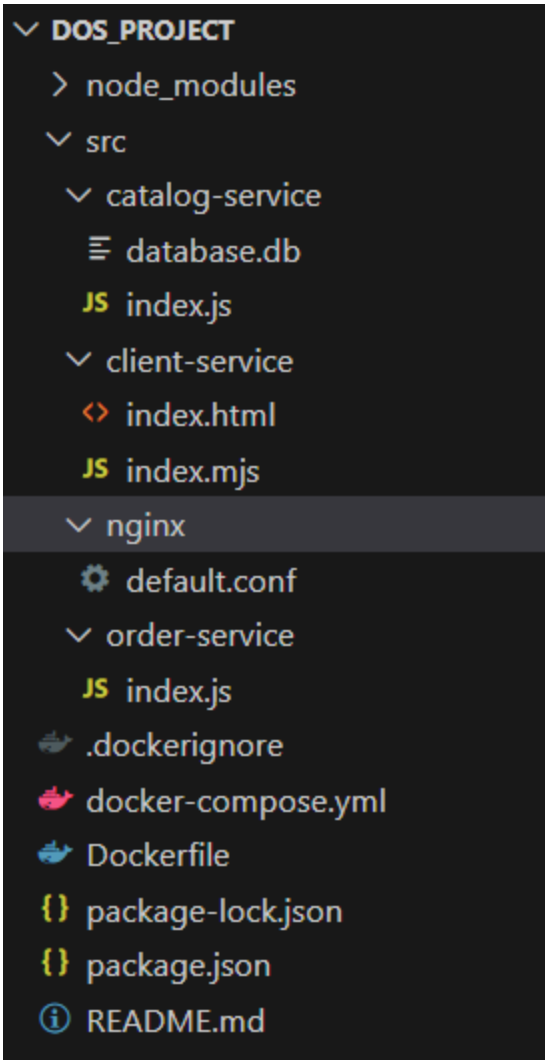
DOS Project Part #1 Report

Student Name: Abdullah Ghassan Sholi

Student Name: Saif Al Dawla Abbas

Student Name: Osaid Ahmed Yaseen

- In the first, this our `project Hierarchy`:



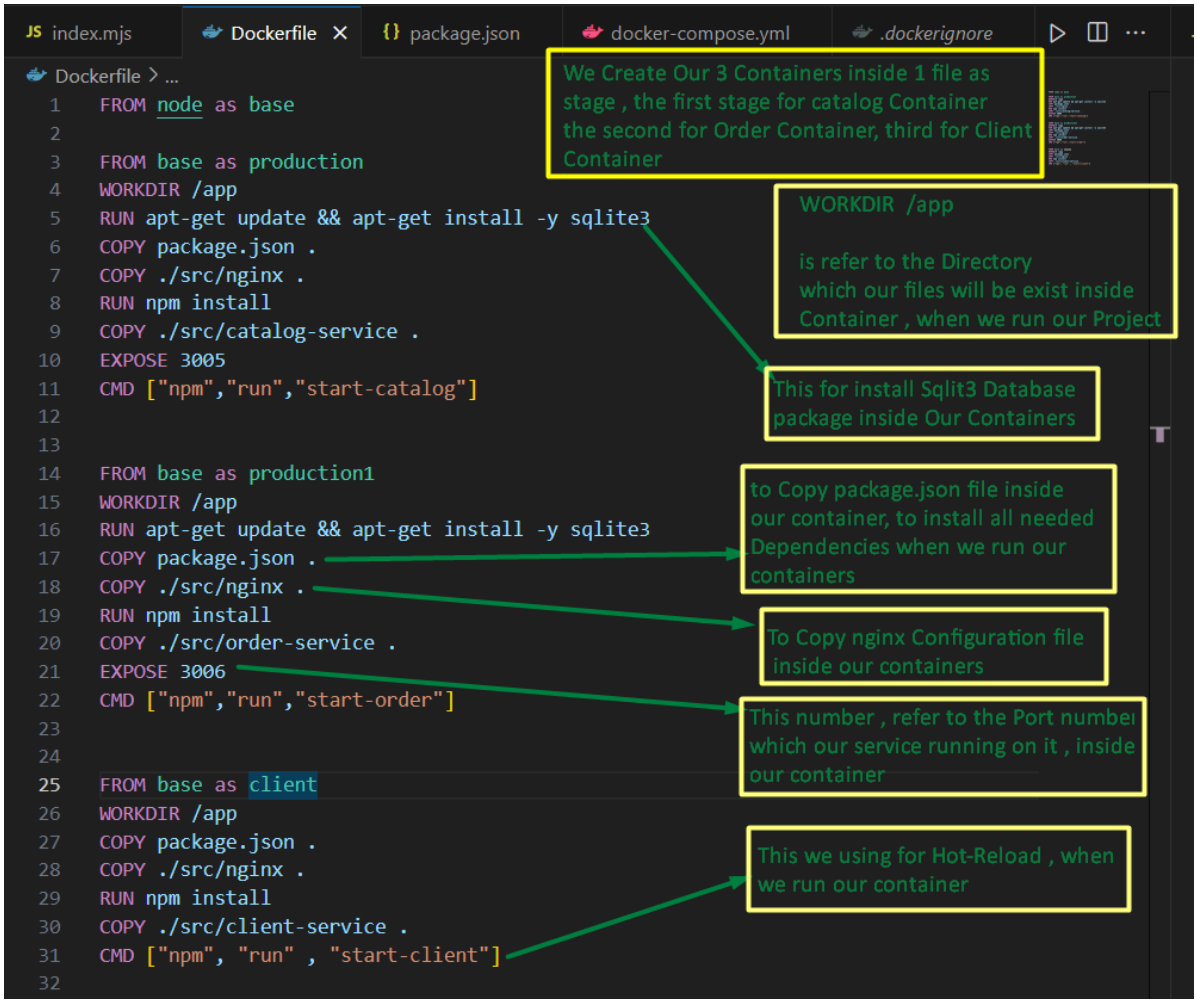
- we create 3 Services , 2 for Backend servers , Catalog & Order,

1 for Frontend Client Service.

- we create `Dockerfile`, to create our containers.
- we create `container for each service` , so we use `docker-compose` tool.
  - docker compose tool: is a tool for definig and running multi-container Docker Applications.
  - so we create docker-compose.yml to write the configuration for our compose files
- to `communication between services` and treatment with incoming traffic from multiple servers , so we using `nginx Load balancing.`
  - so we create `default.conf` file to type the configurations for nginx

---

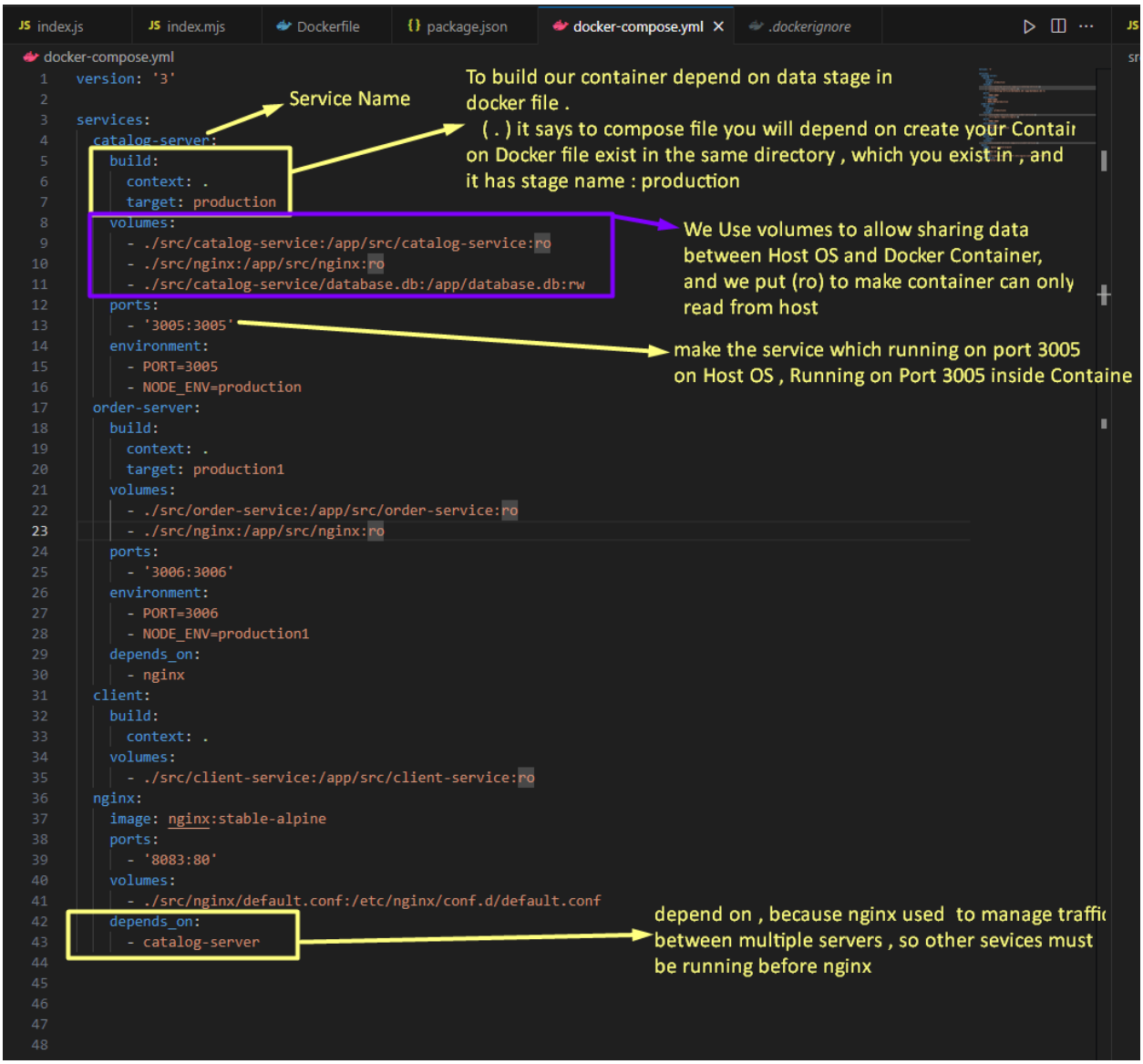
- Now, Lets explain each part of our code:
  -



- inside our `package.json` file:
  - we used nodemon package for Hot-Reload :

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start-catalog": "nodemon -L src/catalog-service/index.js",
  "start-order": "nodemon -L src/order-service/index.js",
  "start-client": "nodemon -L src/client-service/index.mjs"
},
```

- `Docker-compose.yml` file:



- Services Files
  - client-service —> index.mjs
  - catalog-service —> index.js
  - order-service —> index.js
- catalog-service
  - it contains 3 Requests
    - search by book topic

        - `localhost:3005/search/:bookTopic`
    - info by item number
        - `localhost:3005/info/:itemNumber`
    - purchase book
        - This for recieve request to purchase from order service , to pay a book
- order-service
    - purchase request
        - `localhost:3006/purch`
        - recieve item number & order cost from CLI (frontend) then send them to catalog service using axios post request

```
app.post("/purch",async (req,res)=>{

  const order = {
    "id":req.body.id,
    "orderCost":req.body.orderCost
  };
  try{
    const response = await axios.post(`http://catalog-server:3005/order`,order);
    console.log(response.data)

    res.send({message:"Send Request To Catalog"})
  } catch(err){
    console.log(err)
    res.status(400).send({error:err})
  }
})
```

        - then in it fetch the order request in catalog server ,
            - in the first search about the book by id in database
            - then check if amount of money you paid is sufficient to pay the book or not , if yes , decrease number of items on the stock
            - then send message you bought book “ book name ”
            - if book not exist or not enter enough money to pay , send response failed to buy the book

- the response send to the order server in the last operation .

- client-service
  - i used `commander & inquirer` package to create my CLI tool
  - i used `axios` package to fetch backend server requests
  - Here is a `guide` to run my project from CLI

```
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs
Usage: CLI [options] [command]

CLI for DOS Project

Options:
  -V, --version                  output the version number
  -h, --help                     display help for command

Commands:
  search-book-title|s            search about specific book using book topic
  info-book-item-number|i        info about specific book using item number
  purchase-book-by-item-number|p purchase specific book using item number
  help [command]                 display help for command
```

- now, running the project step by step :

To run our containers :

```
docker-compose up -d –build    --> to build all containers
docker-compose down            --> to stop all containers
```

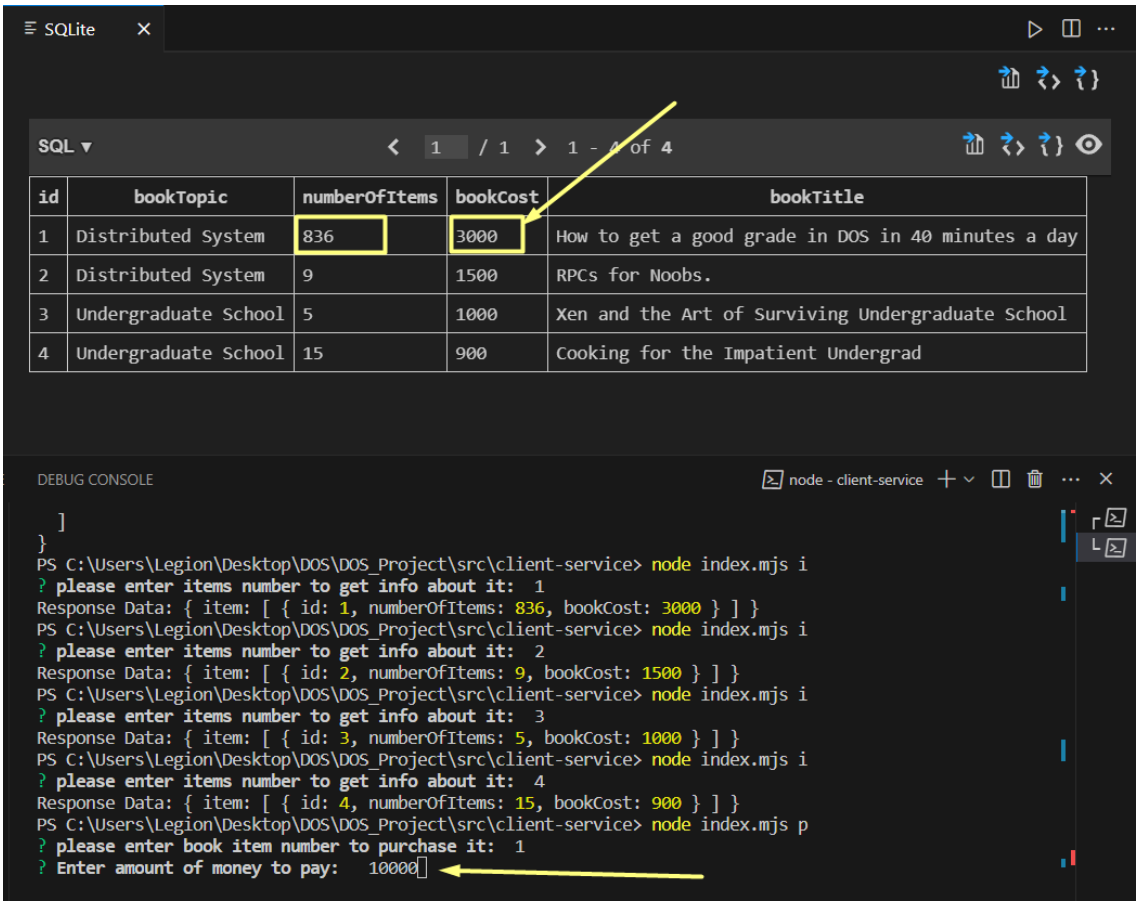
- search request:

```
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs s
? please enter book topic to get details about it:  Distributed System
Response Data: {
  items: [
    {
      id: 1,
      bookTopic: 'Distributed System',
      numberOfItems: 836,
      bookCost: 3000,
      bookTitle: 'How to get a good grade in DOS in 40 minutes a day'
    },
    {
      id: 2,
      bookTopic: 'Distributed System',
      numberOfItems: 9,
      bookCost: 1500,
      bookTitle: 'RPCs for Noobs.'
    }
  ]
}
```

- info request:

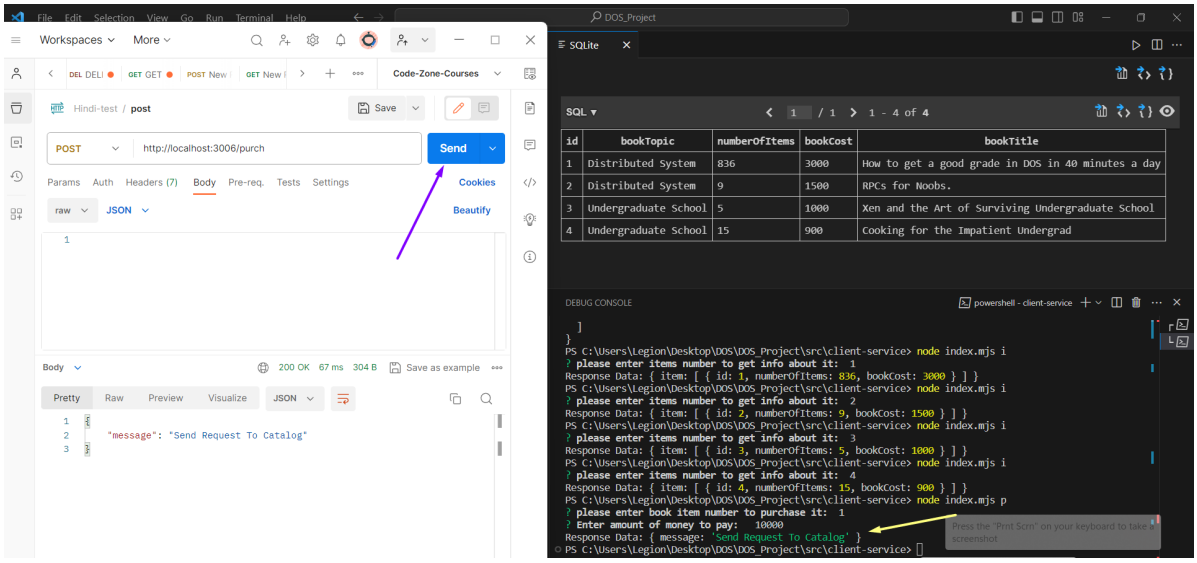
```
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs i
? please enter items number to get info about it:  1
Response Data: { item: [ { id: 1, numberOfItems: 836, bookCost: 3000 } ] }
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs i
? please enter items number to get info about it:  2
Response Data: { item: [ { id: 2, numberOfItems: 9, bookCost: 1500 } ] }
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs i
? please enter items number to get info about it:  3
Response Data: { item: [ { id: 3, numberOfItems: 5, bookCost: 1000 } ] }
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> node index.mjs i
? please enter items number to get info about it:  4
Response Data: { item: [ { id: 4, numberOfItems: 15, bookCost: 900 } ] }
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service>
```

- purchase request:
  - step 1:

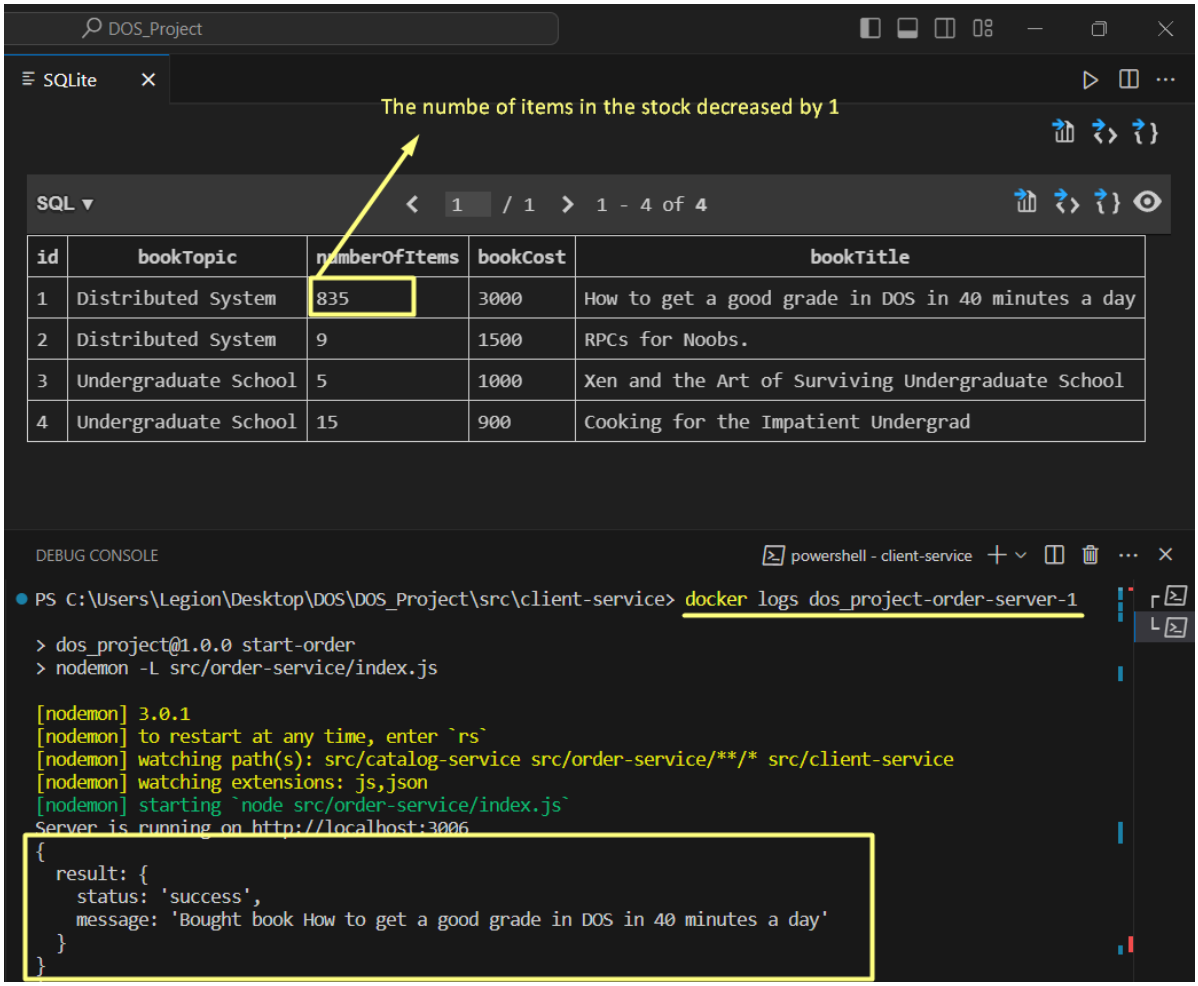


type enter

- step 2:



- step 3:


DOS_Project
SQLite
The numbe of items in the stock decreased by 1
SQL
1 / 1
1 - 4 of 4
id | bookTopic | numberOfItems | bookCost | bookTitle
1 | Distributed System | 835 | 3000 | How to get a good grade in DOS in 40 minutes a day
2 | Distributed System | 9 | 1500 | RPCs for Noobs.
3 | Undergraduate School | 5 | 1000 | Xen and the Art of Surviving Undergraduate School
4 | Undergraduate School | 15 | 900 | Cooking for the Impatient Undergrad
DEBUG CONSOLE
powershell - client-service
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> docker logs dos_project-order-server-1
> dos_project@1.0.0 start-order
> nodemon -L src/order-service/index.js
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src/catalog-service src/order-service/**/* src/client-service
[nodemon] watching extensions: js,json
[nodemon] starting `node src/order-service/index.js`
Server is running on http://localhost:3006
{
result: {
status: 'success',
message: 'Bought book How to get a good grade in DOS in 40 minutes a day'
}
}