

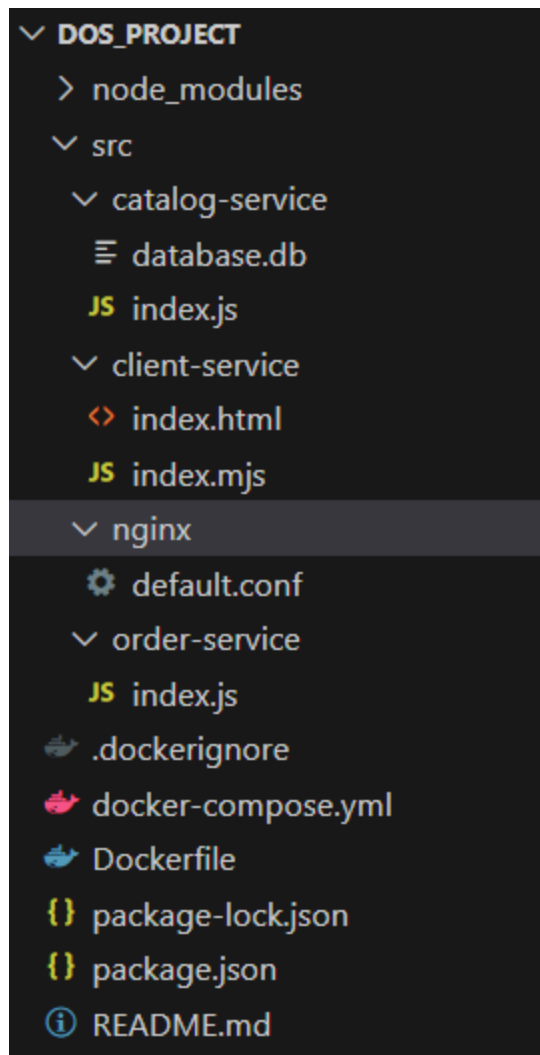
DOS Project Part #1 Report

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:
 -

The image shows a Dockerfile with three stages: `base`, `production1`, and `client`. Annotations explain the purpose of various commands:

- Stage 1 (base):** `FROM node as base` is the first stage for the catalog container.
- WORKDIR /app:** Refers to the directory where files will exist inside the container.
- Line 5:** `RUN apt-get update && apt-get install -y sqlite3` is for installing the SQLite3 database package.
- Line 17:** `COPY package.json .` copies the package.json file to install dependencies.
- Line 18:** `COPY ./src/nginx .` copies the nginx configuration file.
- Line 21:** `EXPOSE 3006` refers to the port number the service runs on.
- Line 31:** `CMD ["npm", "run", "start-client"]` is used for hot-reload when running the container.

```
1 FROM node as base
2
3 FROM base as production
4 WORKDIR /app
5 RUN apt-get update && apt-get install -y sqlite3
6 COPY package.json .
7 COPY ./src/nginx .
8 RUN npm install
9 COPY ./src/catalog-service .
10 EXPOSE 3005
11 CMD ["npm", "run", "start-catalog"]
12
13
14 FROM base as production1
15 WORKDIR /app
16 RUN apt-get update && apt-get install -y sqlite3
17 COPY package.json .
18 COPY ./src/nginx .
19 RUN npm install
20 COPY ./src/order-service .
21 EXPOSE 3006
22 CMD ["npm", "run", "start-order"]
23
24
25 FROM base as client
26 WORKDIR /app
27 COPY package.json .
28 COPY ./src/nginx .
29 RUN npm install
30 COPY ./src/client-service .
31 CMD ["npm", "run", "start-client"]
32
```

- 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:

```
1 version: '3'
2
3 services:
4   catalog-service:
5     build:
6       context: .
7       target: production
8     volumes:
9       - ./src/catalog-service:/app/src/catalog-service:ro
10      - ./src/nginx:/app/src/nginx:ro
11      - ./src/catalog-service/database.db:/app/database.db:rw
12     ports:
13       - '3005:3005'
14     environment:
15       - PORT=3005
16       - NODE_ENV=production
17   order-service:
18     build:
19       context: .
20       target: production1
21     volumes:
22       - ./src/order-service:/app/src/order-service:ro
23       - ./src/nginx:/app/src/nginx:ro
24     ports:
25       - '3006:3006'
26     environment:
27       - PORT=3006
28       - NODE_ENV=production1
29     depends_on:
30       - nginx
31   client:
32     build:
33       context: .
34     volumes:
35       - ./src/client-service:/app/src/client-service:ro
36   nginx:
37     image: nginx:stable-alpine
38     ports:
39       - '8083:80'
40     volumes:
41       - ./src/nginx/default.conf:/etc/nginx/conf.d/default.conf
42     depends_on:
43       - catalog-service
```

Service Name (points to `catalog-service`)

To build our container depend on data stage in docker file .
(.) it says to compose file you will depend on create your Contain on Docker file exist in the same directory , which you exist in , and it has stage name : production

We Use volumes to allow sharing data between Host OS and Docker Container, and we put (ro) to make container can only read from host

make the service which running on port 3005 on Host OS , Running on Port 3005 inside Containe

depend on , because nginx used to manage traffic between multiple servers , so other services must be running before nginx

- 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 receive request to purchase from order service , to pay a book
- order-service
 - purchase request
 - `localhost:3006/purch`
 - receive 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:

SQLite

SQL ▾ < 1 / 1 > 1 - 4 of 4

id	bookTopic	numberOfItems	bookCost	bookTitle
1	Distributed System	836	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 node - client-service

```

}
}
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> node index.mjs p
? please enter book item number to purchase it: 1
? Enter amount of money to pay: 10000

```

type enter

- step 2:

Workspaces More ▾

POST http://localhost:3006/purch Send

Params Auth Headers (7) Body Pre-req. Tests Settings

raw JSON Beautify

Body

```

1
2
3
"message": "Send Request To Catalog"

```

200 OK 67 ms 304 B Save as example

DEBUG CONSOLE powershell - client-service

```

}
}
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> node index.mjs p
? please enter book item number to purchase it: 1
? Enter amount of money to pay: 10000
Response Data: { message: "Send Request To Catalog" }
PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service>

```


- step 3:

The screenshot shows a web application interface for a SQLite database. At the top, a yellow text message states: "The numbe of items in the stock decreased by 1". Below this is a table with 5 columns: id, bookTopic, numberOfItems, bookCost, and bookTitle. The table contains 4 rows of data. The value '835' in the 'numberOfItems' column of the first row is highlighted with a yellow box, and a yellow arrow points from the text above to this box. Below the table is a 'DEBUG CONSOLE' section showing terminal output. The first line of the terminal is highlighted in yellow: `PS C:\Users\Legion\Desktop\DOS\DOS_Project\src\client-service> docker logs dos_project-order-server-1`. The subsequent lines show the output of the command, including the start of the 'order-service' and a JSON response.

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 + v [] [] ... x

• 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'
  }
}
```