



DOS Project Report Part 2

Student Name: Abdullah Ghassan Sholi Stu#: 12027918

Student Name: Safi Al-Dawla Abbas

Student Name: Osaid Ahmed Yaseen

Part1: Cache Consistency

```

1  app.get('/info/:id', async (req, res) => {
2    let id = req.params.id;
3    console.log(id);
4    const cachedPost = await client.get(`${id}`)
5    // console.log(cachedPost)
6    ///////////////////////////////////////////////////
7    db.serialize(() => {
8      // i used serialize to solve of close data base for data displayed completely
9      db.all(`SELECT id,numberOfItems,bookCost FROM items WHERE id=${id}`, async (err, row) => {
10        if (err) {
11          console.log(err);
12          return;
13        }
14        if(cachedPost){
15          let temp = JSON.parse(cachedPost)
16          console.log(row[0].numberOfItems,"--")
17          console.log(temp.numberOfItems,"--")
18          if(row[0].numberOfItems == temp.numberOfItems)
19            return res.json(JSON.parse(cachedPost))
20          else{
21            client.del(`${id}`)
22            return res.json({Message:"Invalidate"})
23          }
24        }
25        client.set(`${id}`,JSON.stringify(row[0]))
26        console.log(row);
27        res.json({item:row});
28      });
29    });
30  });

```

When Data Exist in Cache Memory

When Data in Cache different from data in Original Database

When Data Comming from original Database

- When i send **GET** request the first time, before Caching the data:

Search Postman

Invite Upgrade

Code-Zone-Courses

Hindi-test / search book

GET http://localhost:3005/info/1

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

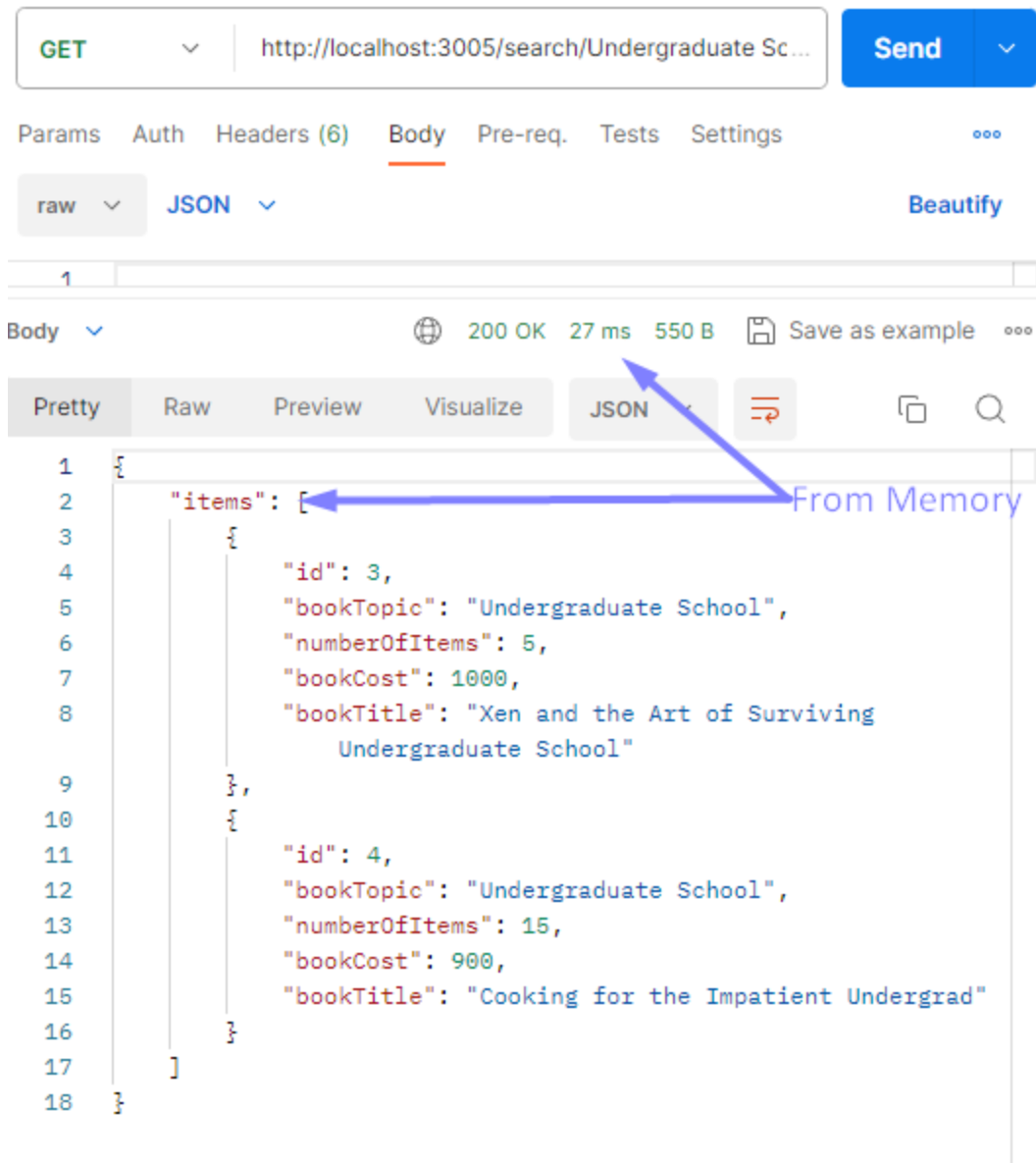
1

Body Cookies Headers (8) Test Results

200 OK 456 ms 322 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "item": [
3     {
4       "id": 1,
5       "numberOfItems": 812,
6       "bookCost": 3000
7     }
8   ]
9 }
```



Q1) Compute the average response time (query/buy) of your new systems.
What is the response time with and without caching?

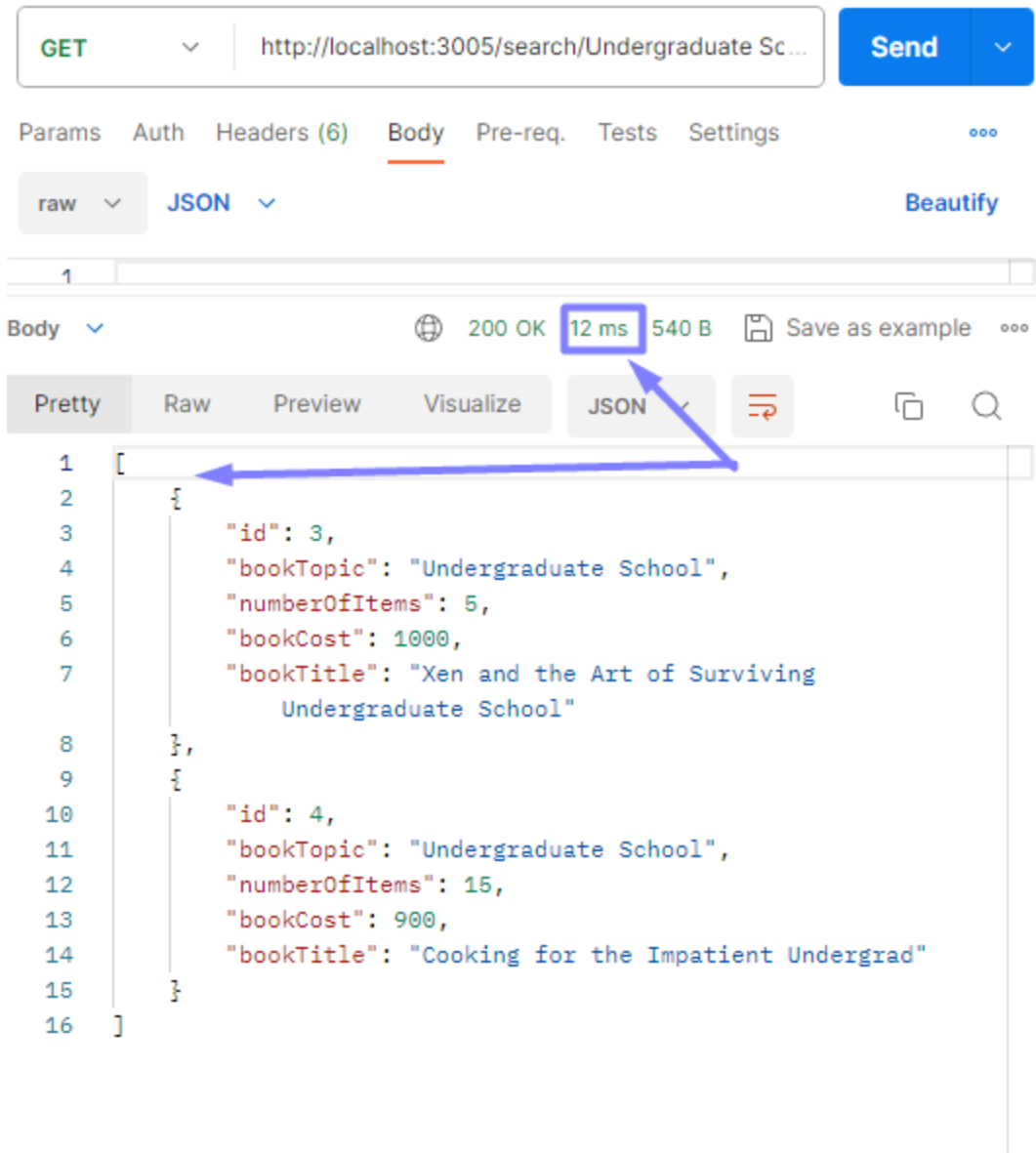
- Answers
 - for info: **456ms**
 - for search: **27ms**

- With Cache:

The screenshot displays the Postman interface for a REST client. At the top, there's a search bar and navigation icons. The main area shows a GET request to `http://localhost:3005/info/1`. Below the URL bar, tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings are visible. The Body tab is selected, showing a JSON body with the following content:

```
1 {
2   "id": 1,
3   "numberOfItems": 812,
4   "bookCost": 3000
5 }
```

At the bottom, the response status is `200 OK` with a response time of `41 ms` (highlighted with a red box) and a response size of `311 B`. The response body is also shown in a JSON format, matching the request body.



Q2) How much does caching help?

- Answers
 - for info: **41ms** , **456/41** —> **11.12** Faster than without cache
 - for search: **12ms**, **27/12** —> **2.25** Faaster than without using cache

Invalidate Message

- for Original services:

The screenshot displays a development environment for a DOS Project Part 2. A REST client window is open, showing a GET request to `http://localhost:3005/info/1` with a response body of `{ "Message": "Invalidate" }`. The source code in the background shows a Node.js application with a `catalog-service` and `index.js` file. The terminal output shows the following sequence of events:

```

PS C:\Users\Ulgion\Desktop\Test Dos\DOS_Project_Part2\src\client-service> node index.mjs 1
? please enter items number to get info about it: 1
Response Data: { id: 1, numberOfItems: 800, bookCost: 3000 }
11 From Replica
PS C:\Users\Ulgion\Desktop\Test Dos\DOS_Project_Part2\src\client-service> node index.mjs 1
? please enter items number to get info about it: 1
Response Data: { id: 1, numberOfItems: 801, bookCost: 3000 }
11 From Replica
PS C:\Users\Ulgion\Desktop\Test Dos\DOS_Project_Part2\src\client-service> node index.mjs p
? please enter book item number to purchase it: 1
? Enter amount of money to pay: 11111111111111111111
Response Data: { message: "Send Request To Catalog" }
11 From Original
PS C:\Users\Ulgion\Desktop\Test Dos\DOS_Project_Part2\src\client-service>
  
```

The REST client window also shows a table of book topics:

id	bookTopic
1	Distributed System
2	Distributed System
3	Undergraduate School
4	Undergraduate School

The image shows a VS Code editor with an HTTP client extension. The request is a GET to `http://localhost:3005/info/1`. The response is a 200 OK with a 30 ms duration and a JSON body containing book details. Below the VS Code window is a terminal window showing the output of a command, with a box highlighting the values 1, 800, and 800.

VS Code HTTP Client Request:

- Method: GET
- URL: `http://localhost:3005/info/1`
- Body:

```
{
  "id": 1,
  "numberOfItems": 800,
  "bookCost": 3000
}
```

VS Code HTTP Client Response:

- Status: 200 OK
- Duration: 30 ms
- Size: 311 B
- Body:

```
{
  "id": 1,
  "numberOfItems": 800,
  "bookCost": 3000
}
```

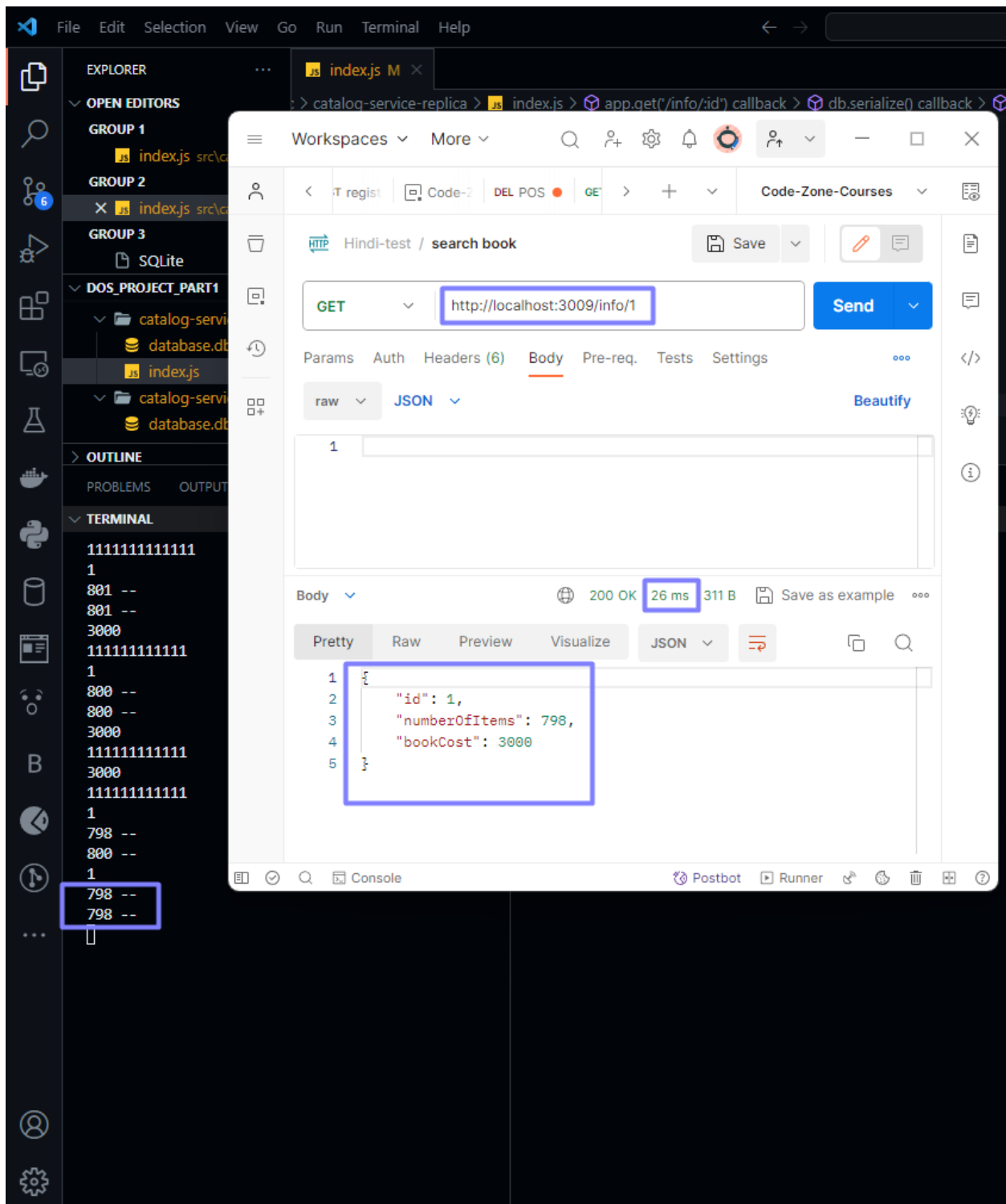
Terminal Output:

```
3000
11111111111111
1
801 --
802 --
1
801 --
801 --
3000
11111111111111
1
800 --
801 --
1
800 --
800 --
```


- for Replica Service

The screenshot displays the VS Code interface with three main panels illustrating the replica service workflow:

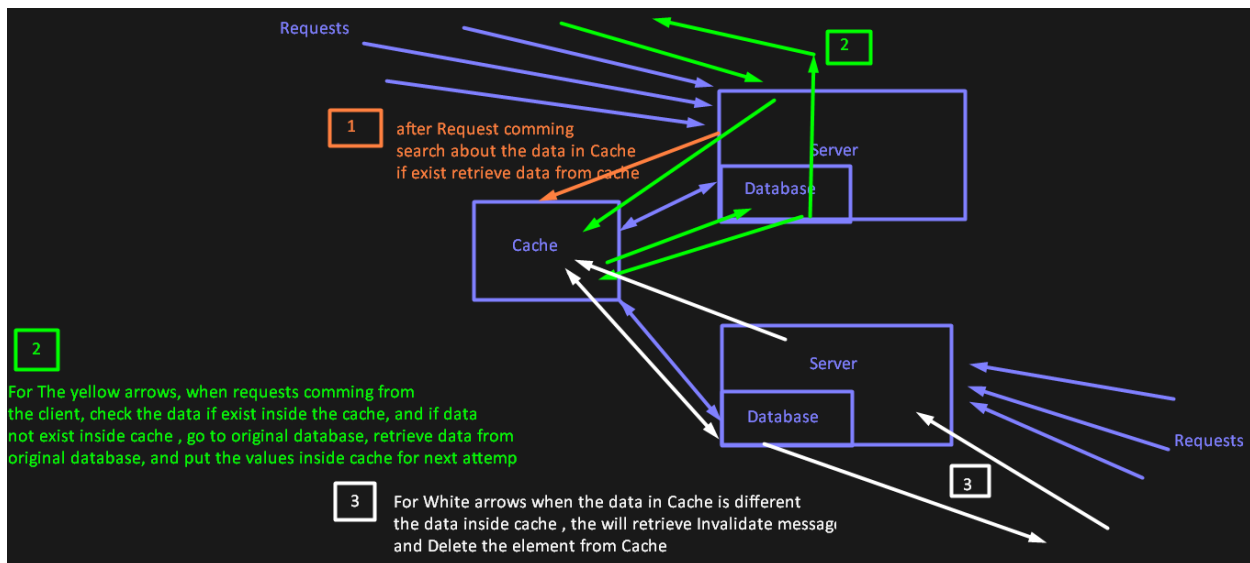
- REST Client Panel (Left):** Shows a GET request to `http://localhost:3009/info/1`. The response body is a JSON object: `{ "Message": "Invalidate" }`. A blue arrow points from the URL to the terminal.
- Source Code Panel (Top Right):** Shows the `index.js` file with a query: `db.all('SELECT id,numberOfItems,bookCost FROM Items WHERE id=?', async (err) { ... })`. Annotations highlight `Data In Original Database` and `Data In Cache Memory`. A blue arrow points from the `db.all` call to the terminal.
- Terminal Panel (Bottom):** Shows the command prompt output for the `node index.js` command. It displays the sequence of events:
 - 1. Initial request and response.
 - 2. Subsequent request and response.
 - 3. Final request and response.



Expirement Run	Without Cache	With Cache	#of Times when using Cache speed
----------------	---------------	------------	----------------------------------

1	97	34	$97/34 = 2.853$ Times
2	29	17	$29/17 = 1.71$ Times
3	40	14	$40/14 = 2.857$ Times

The Procedure:



Part2: Loadbalance with NGINX

I Used Nginx to achieve loadbalance, each service exists in its separate Docker Container and it has its own Interface & Port to communicate with other services, Below my File Configuration for NGINX

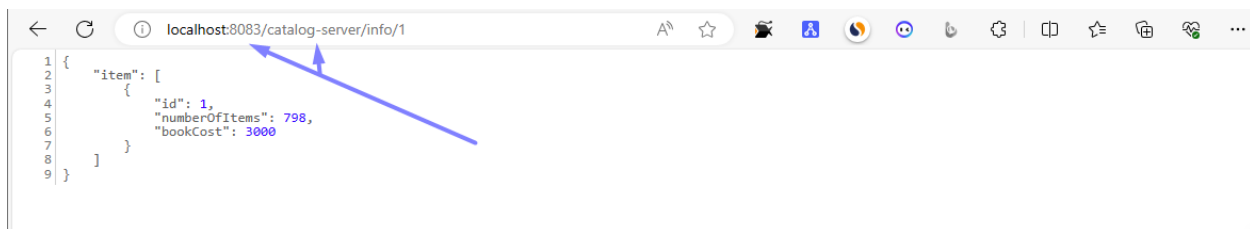
You, 3 days ago | 2 authors (AbdullahSholi and others) | [Click here to ask Blackbox to help you code faster](#)

```

1  upstream catalog-server {
2      server catalog-server:3005;
3  }
4
5  upstream catalog-server-replica {
6      server catalog-server-replica:3009;
7  }
8
9
10 upstream order-server {
11     server order-server:3006;
12 }
13
14 upstream order-server-replica {
15     server order-server-replica:3008;
16 }
17
18 upstream client {
19     server client:3007;
20 }
21
22 server { # simple reverse-proxy
23     listen      80;
24     # port 80 for http      AbdullahSholi, last month • Whole Project Files
25     # port 443 for https
26     # بما يبدأ اينجن اكس يبعث ريكويستات على الويبسايت تبقي
27     # يعمل الكونفيجريشن اللي تحت عليه
28
29     # / معناها الويبسايت تبينا
30     location / {
31         proxy_set_header Host $host;
32         # enable WebSockets (for ws://sockjs not connected error in the client source: https://stack
33         proxy_http_version 1.1;
34         proxy_set_header Upgrade $http_upgrade;
35         proxy_set_header Connection "upgrade";
36         proxy_pass http://client/;
37     }
38
39     location /catalog-server {
40         rewrite ^/catalog-server/(.*) /$1 break;
41         proxy_set_header X-Real-IP $remote_addr;
42         proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
43         proxy_set_header Host $http_host;
44         proxy_set_header X-NginX-Proxy true;
45     }

```

for testing all container running correctly via nginx:



```
1 {
2   "id": 1,
3   "numberOfItems": 798,
4   "bookCost": 3000
5 }
```

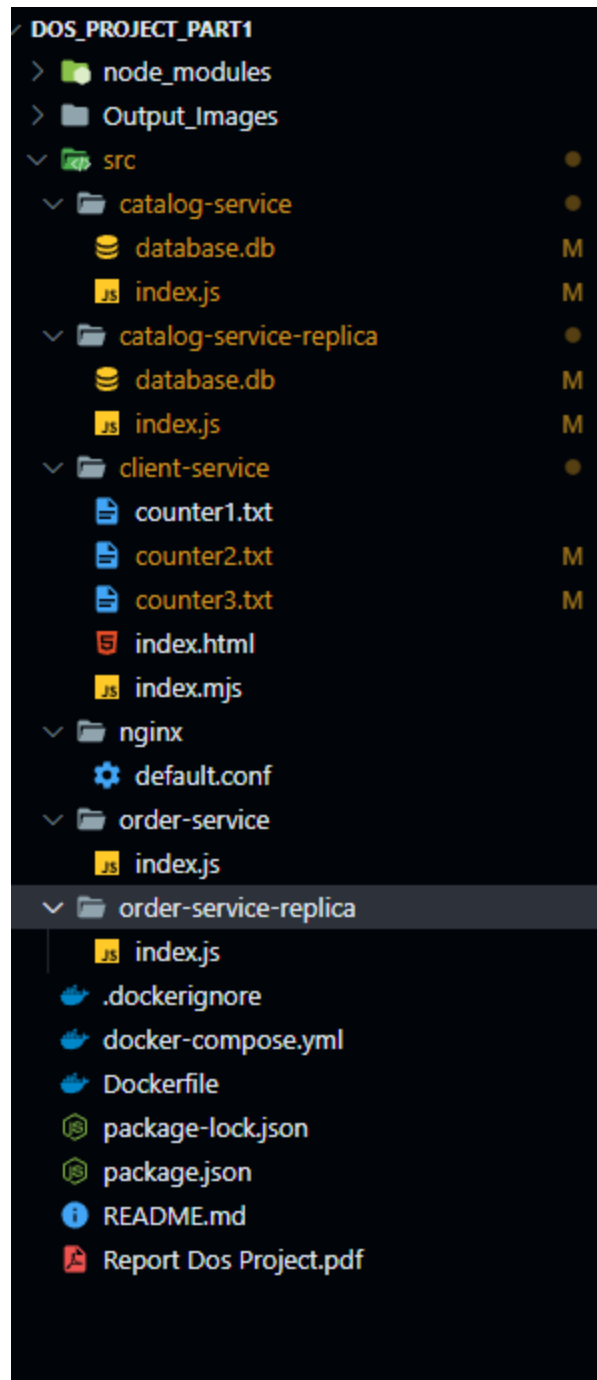
```
1 {
2   "Message": "Arrive"
3 }
```

```
1 {
2   "Message": "Arrive"
3 }
```

```
166 if (err) {
167   console.log(err);
168   return;
169 }
170 if (cached) {
171   let temp = res.json(res.data);
172   console.log(temp);
173   return temp;
174 }
175 if (row) {
176   return res.json(row);
177 } else {
178   client.get('/info/' + itemNumber)
179     .then((res) => {
180       return res.json(res.data);
181     })
182     .catch((err) => {
183       console.log(err);
184       return res.json(res.data);
185     });
186 }
187 res.json(res.data);
188 });
189 app.get('/posts', (req, res) => {
190   const id = req.query.id;
191   console.log(id);
192   const cachedPost = res.json(res.data);
193   if (cachedPost) {
194     return res.json(cachedPost);
195   }
196   const response = client.get('/posts/' + id)
197     .then((res) => {
198       return res.json(res.data);
199     })
200     .catch((err) => {
201       console.log(err);
202       return res.json(res.data);
203     });
204   res.json(response);
205 });
206 app.get('/test', (req, res) => {
207   res.send('Message: Arrive');
208 });
209 app.listen(8083, () => {
210   console.log('Server is running on port 8083');
211 });
```

For Load balance in the frontend side i used counter for each request, the counter for toggle between sending request via Original server_Replica

For Replication Services & Database:



Part3: Dockerize your Application (Optional part)

I Construct my project Dockerized from scratch, i create docker container (image) for each service, and each service has it's own port, and i use volume for sharing data between the Guest OS (Docker Containers) & Host OS , and i create docker-compose file to run all containers at the same time with 1 simple command .

My `docker-compose.yml` file:

```
version: '3'

services:
  catalog-server:
    build:
      context: .
      target: production
    volumes:
      - ./src/catalog-service:/app/src/catalog-service:ro
      - ./src/nginx:/app/src/nginx:ro
      - ./src/catalog-service/database.db:/app/database.db:rw
    ports:
      - '3005:3005'
    environment:
      - PORT=3005
      - NODE_ENV=production
  catalog-server-replica:
    build:
      context: .
      target: production2
    volumes:
      - ./src/catalog-service-replica:/app/src/catalog-service-i
      - ./src/nginx:/app/src/nginx:ro
      - ./src/catalog-service-replica/database.db:/app/database
    ports:
      - '3009:3009'
    environment:
      - PORT=3009
```

```

    - NODE_ENV=production2
order-server:
  build:
    context: .
    target: production1
  volumes:
    - ./src/order-service:/app/src/order-service:ro
    - ./src/nginx:/app/src/nginx:ro
  ports:
    - '3006:3006'
  environment:
    - PORT=3006
    - NODE_ENV=production1
  depends_on:
    - nginx

order-server-replica:
  build:
    context: .
    target: production3
  volumes:
    - ./src/order-service-replica:/app/src/order-service-repl:
    - ./src/nginx:/app/src/nginx:ro
  ports:
    - '3008:3008'
  environment:
    - PORT=3008
    - NODE_ENV=production3
  depends_on:
    - nginx
client:
  build:
    context: .
  volumes:
    - ./src/client-service:/app/src/client-service:ro
nginx:

```



```

image: nginx:stable-alpine
ports:
  - '8083:80'
volumes:
  - ./src/nginx/default.conf:/etc/nginx/conf.d/default.conf
depends_on:
  - catalog-server

redis:
  image: redis

```

Docker Containers(images) while running:

```

service> docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                               NAMES
6735989f1bc   dos_project_part1-order-server      "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 0.0.0.0:3006->3006/tcp              dos_project_part1-order-server-1
4a7beb034c5f   dos_project_part1-order-server-replica "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 0.0.0.0:3008->3008/tcp              dos_project_part1-order-server-replica-1
24500064ec5e   nginx:stable-alpine                 "/docker-entrypoint..." 28 minutes ago Up 28 minutes 0.0.0.0:8083->80/tcp                dos_project_part1-nginx-1
9a9b36630064   dos_project_part1-client            "docker-entrypoint.s..." 28 minutes ago Up 28 minutes                               dos_project_part1-client-1
98dbad981f50   dos_project_part1-catalog-server-replica "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 0.0.0.0:3009->3009/tcp              dos_project_part1-catalog-server-replica-1
d319d44c1118   dos_project_part1-catalog-server     "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 0.0.0.0:3005->3005/tcp              dos_project_part1-catalog-server-1
ae64377che24   redis                                "docker-entrypoint.s..." 28 minutes ago Up 28 minutes 6379/tcp                            dos_project_part1-redis-1
PS C:\Users\Legion\Desktop\Test Dos\DOS_Project_Part1\src\client-service>

```

For Running The Project, it's the same for Part1:

```
docker-compose up -d -- build
```

```
cd src/client-service
```

```
node index.mjs
```

and Here is all others Commands which my Project Support:

```
PS C:\Users\Legion\Desktop\Test Dos\DOS_Project_Part1\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
PS C:\Users\Legion\Desktop\Test Dos\DOS_Project_Part1\src\client-service> 
```