DOS Project Report Part 2

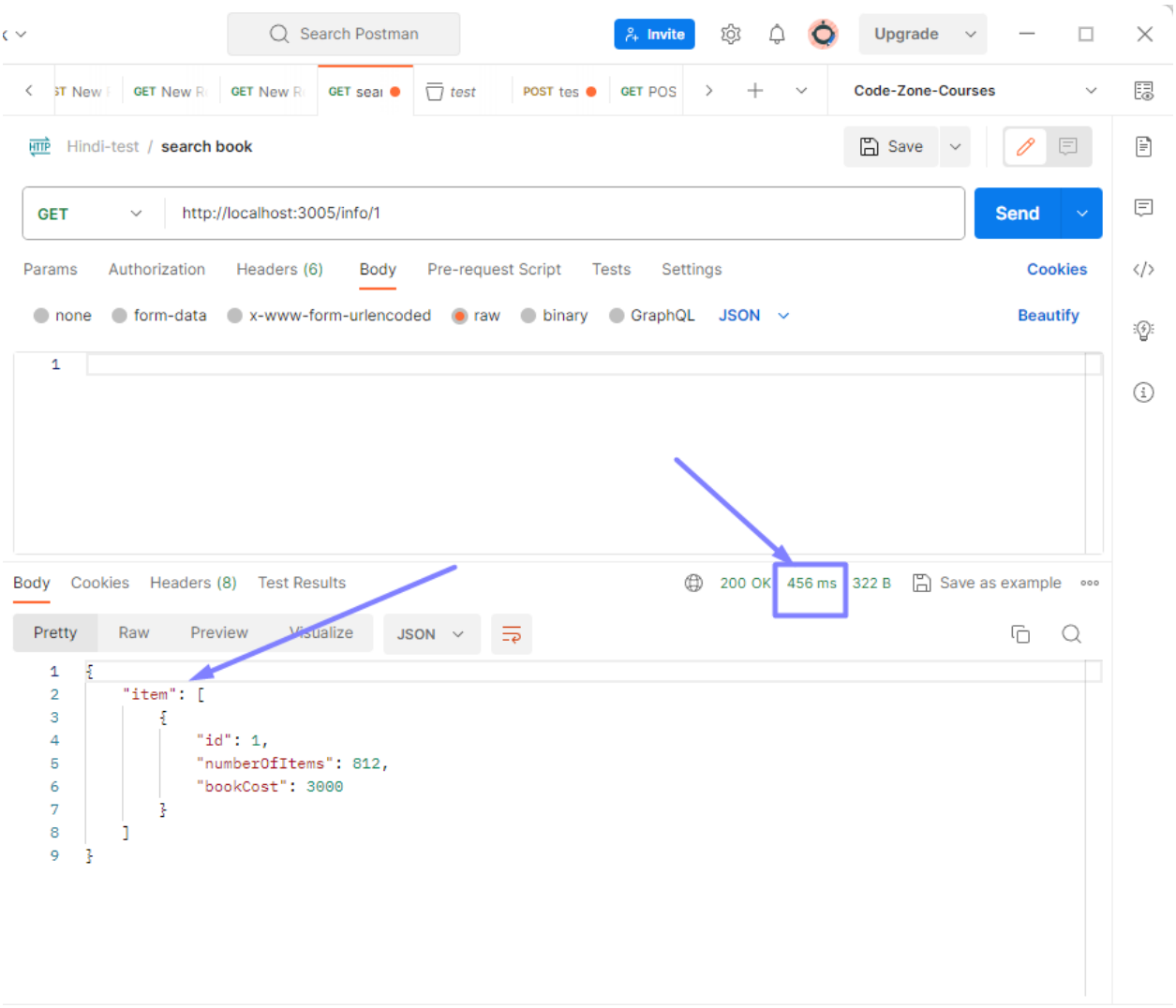
Student Name: Abdullah Ghassan Sholi Stu#: 12027918

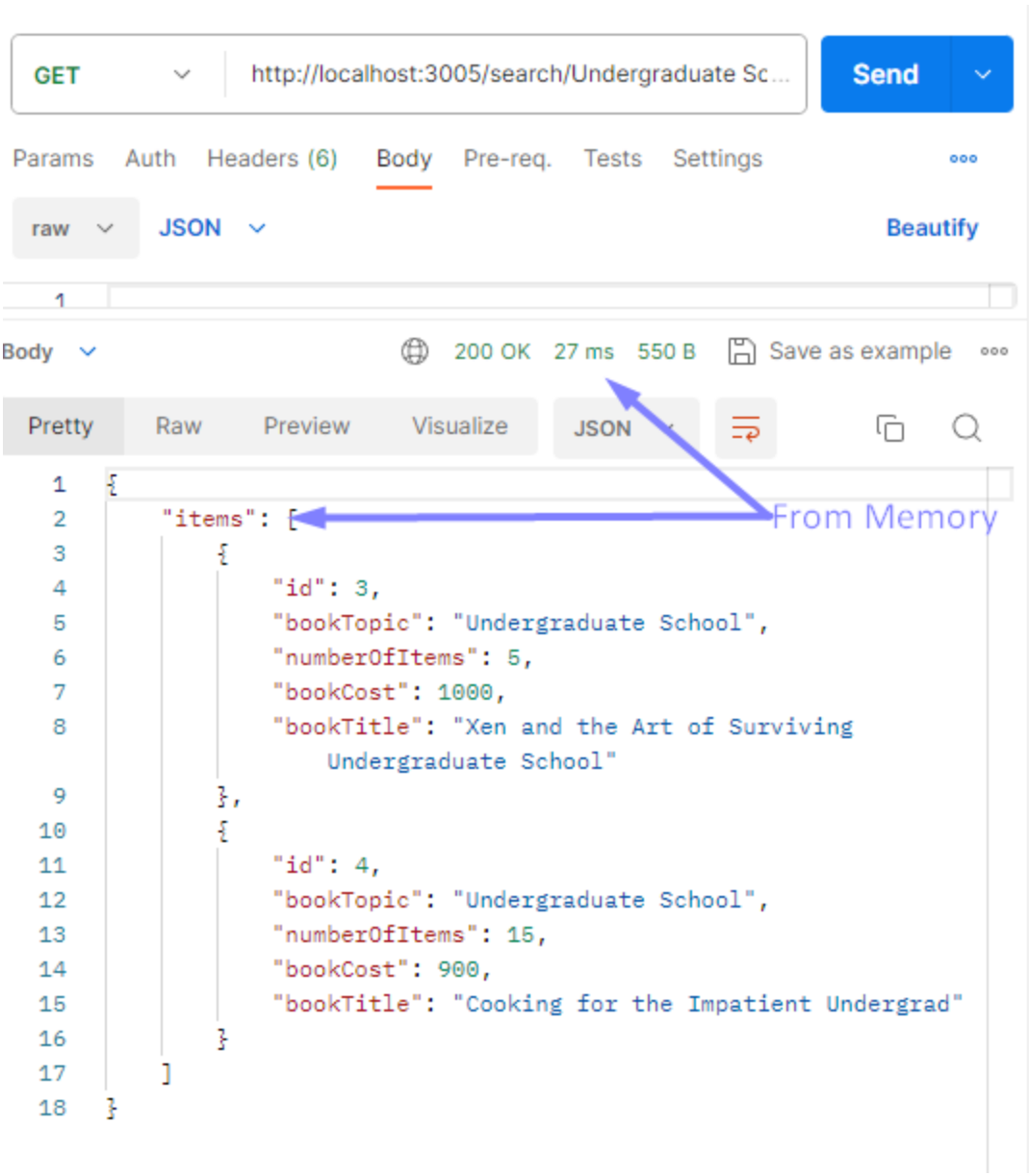
Student Name: Safi Al-Dawla Abbas

Student Name: Osaid Ahmed Yaseen

Part1: Cache Consistency

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

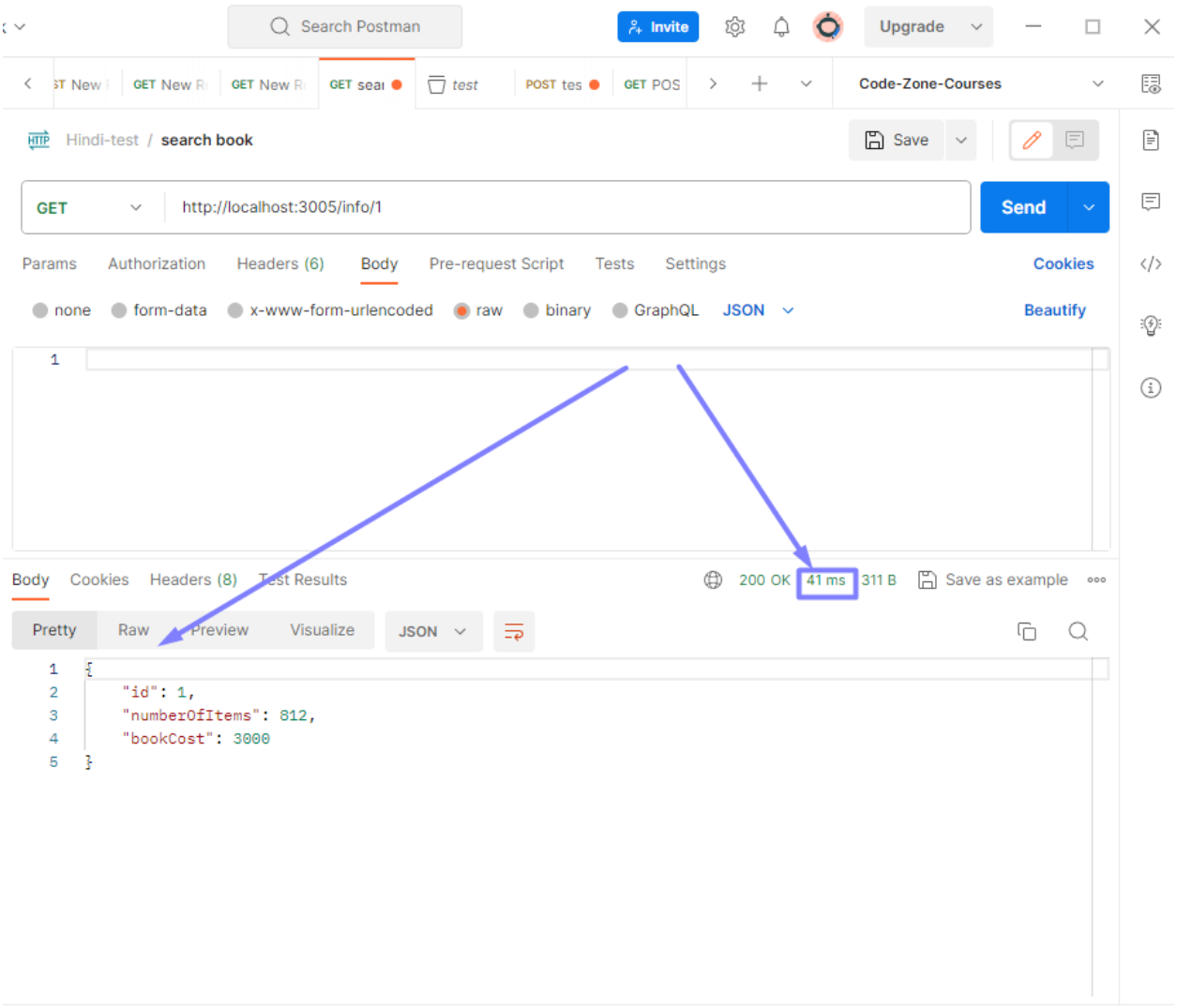

Search Postman
Invite
Upgrade
Code-Zone-Courses
Hindi-test / search book
Save
GET
http://localhost:3005/info/1
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (8)
Test Results
200 OK
456 ms
322 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"item": [
{
"id": 1,
"numberOfItems": 812,
"bookCost": 3000
}
]
}

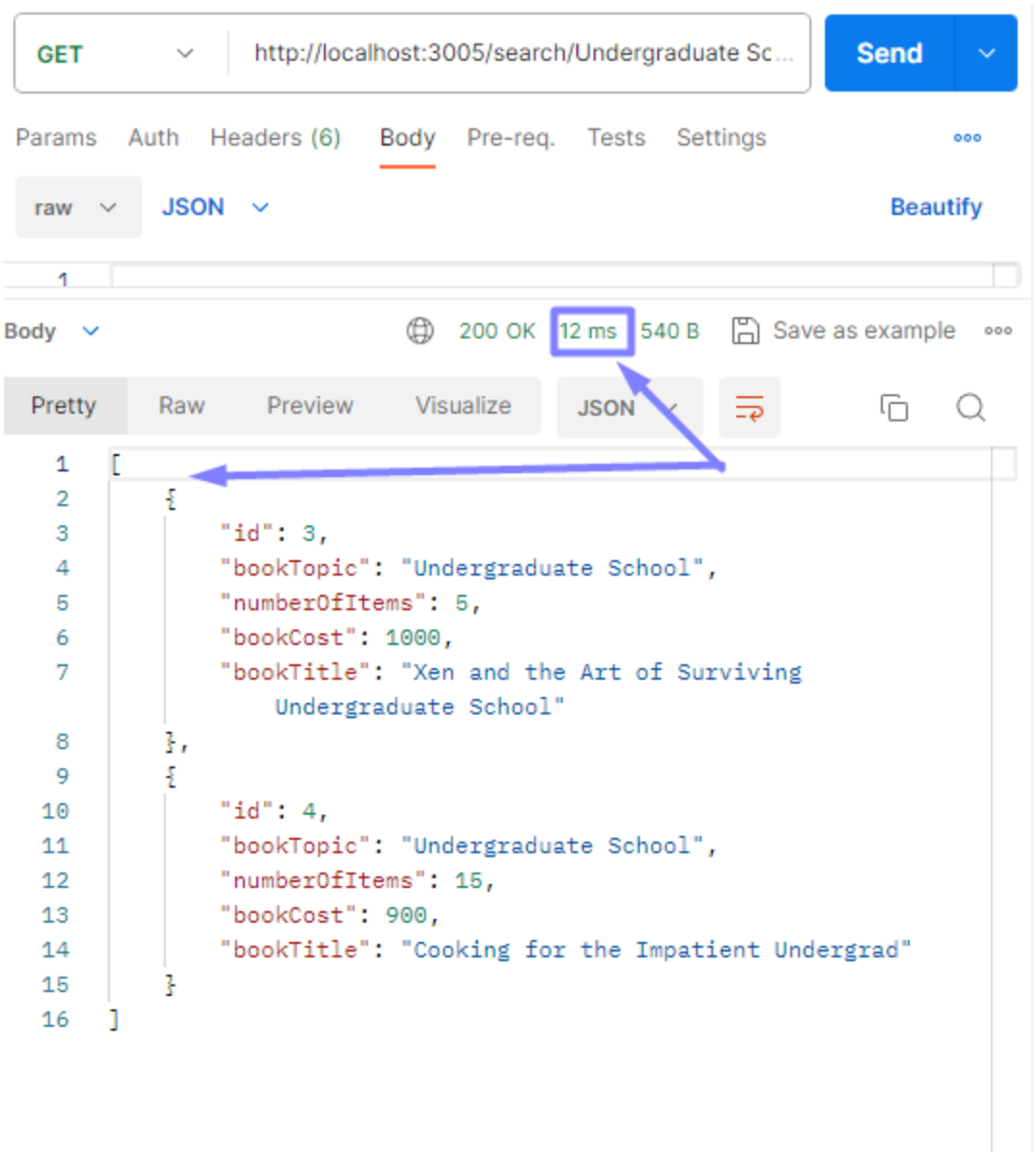


Q1) Compute the average response time (query/buy) of your new systems. What is the response time with and without caching?

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

- With Cache:


Search Postman
Invite
Upgrade
Code-Zone-Courses
Hindi-test / search book
Save
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
Body
Cookies
Headers (8)
Test Results
200 OK
41 ms
311 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"numberOfItems": 812,
"bookCost": 3000
}

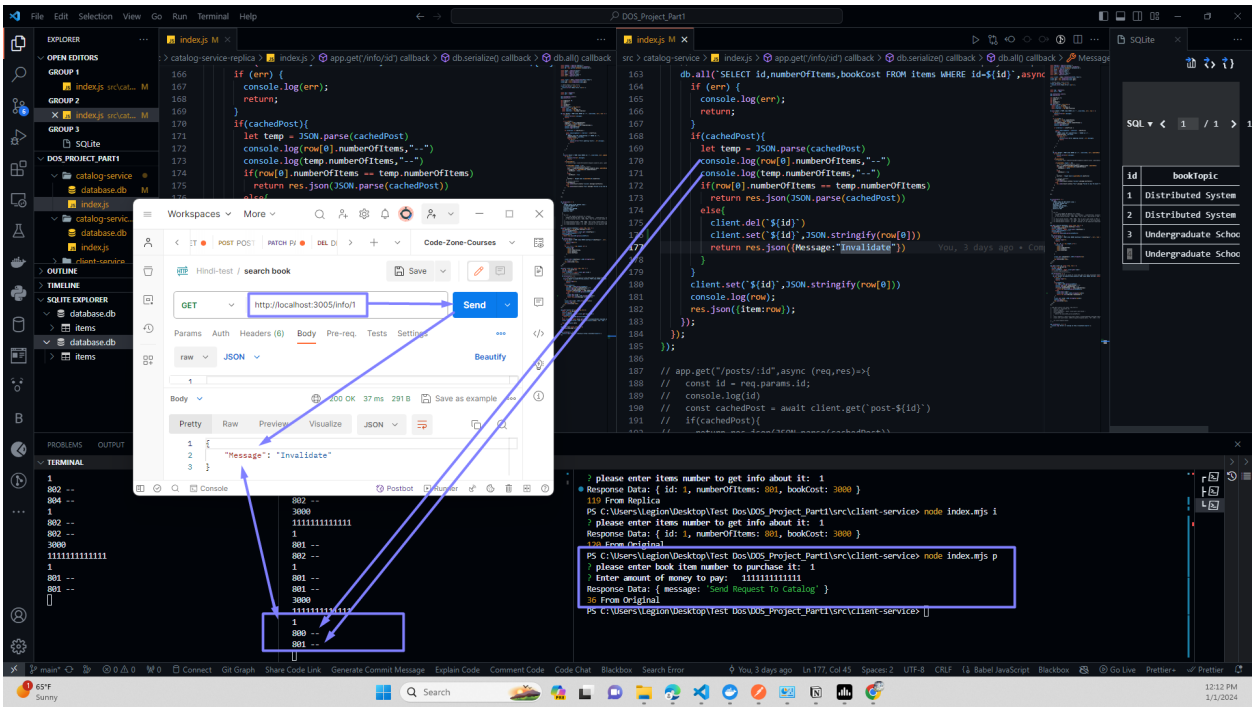


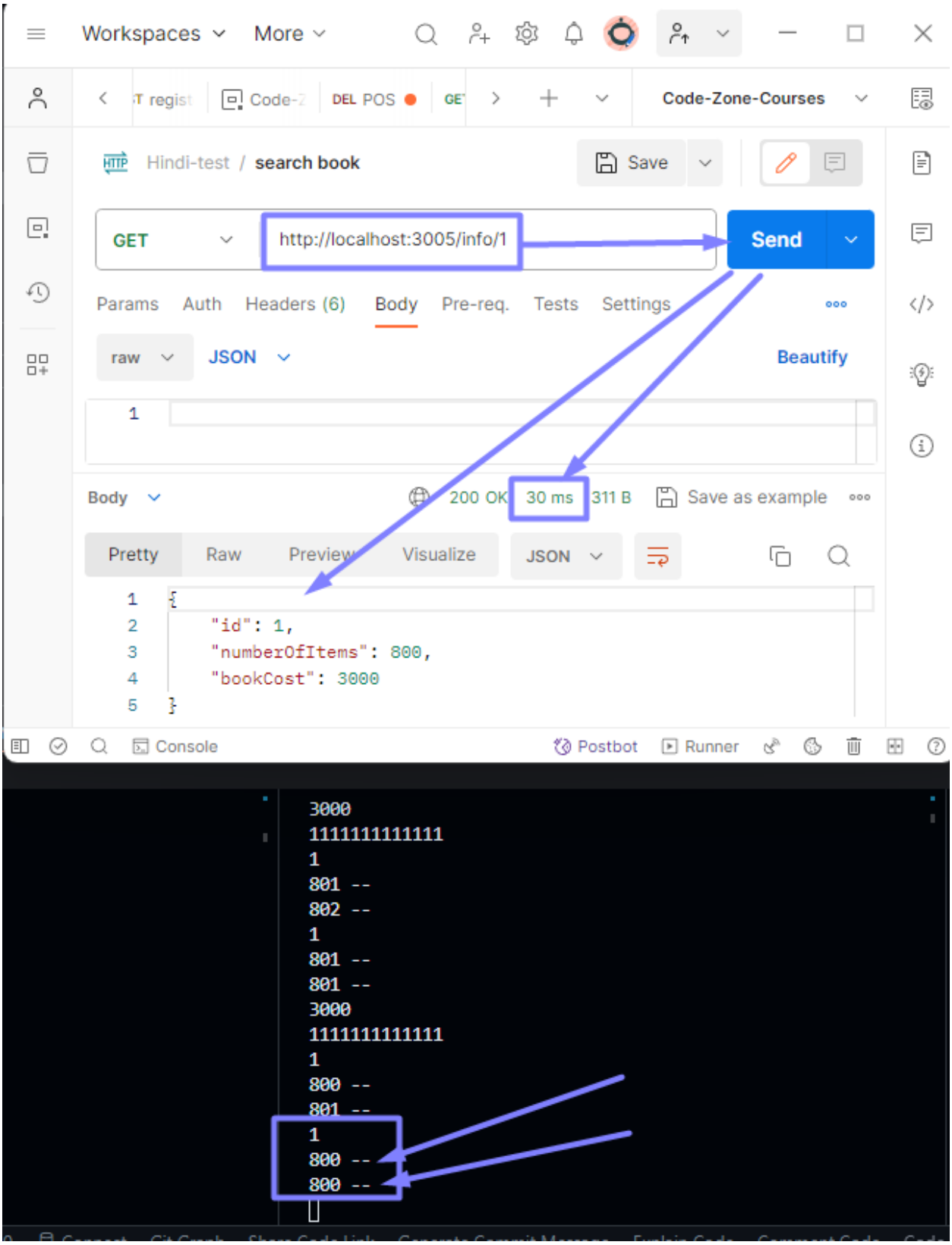
Q2) How much does caching help?

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

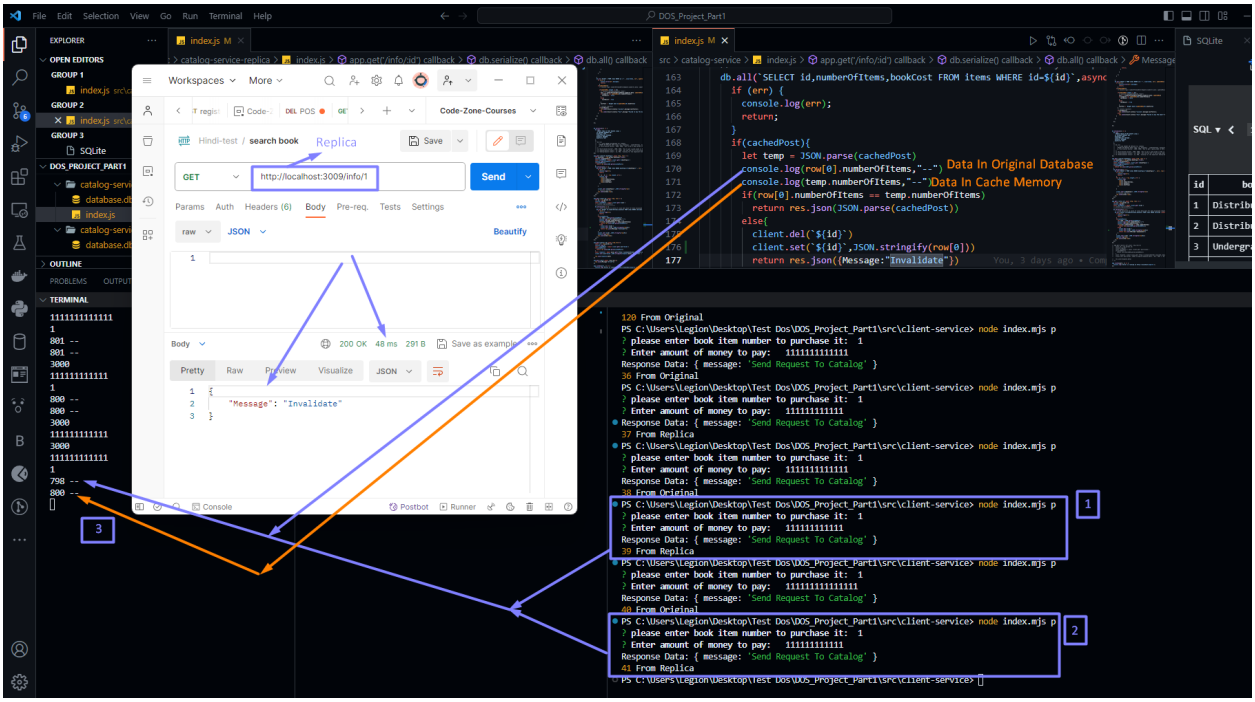
Invalidate Message

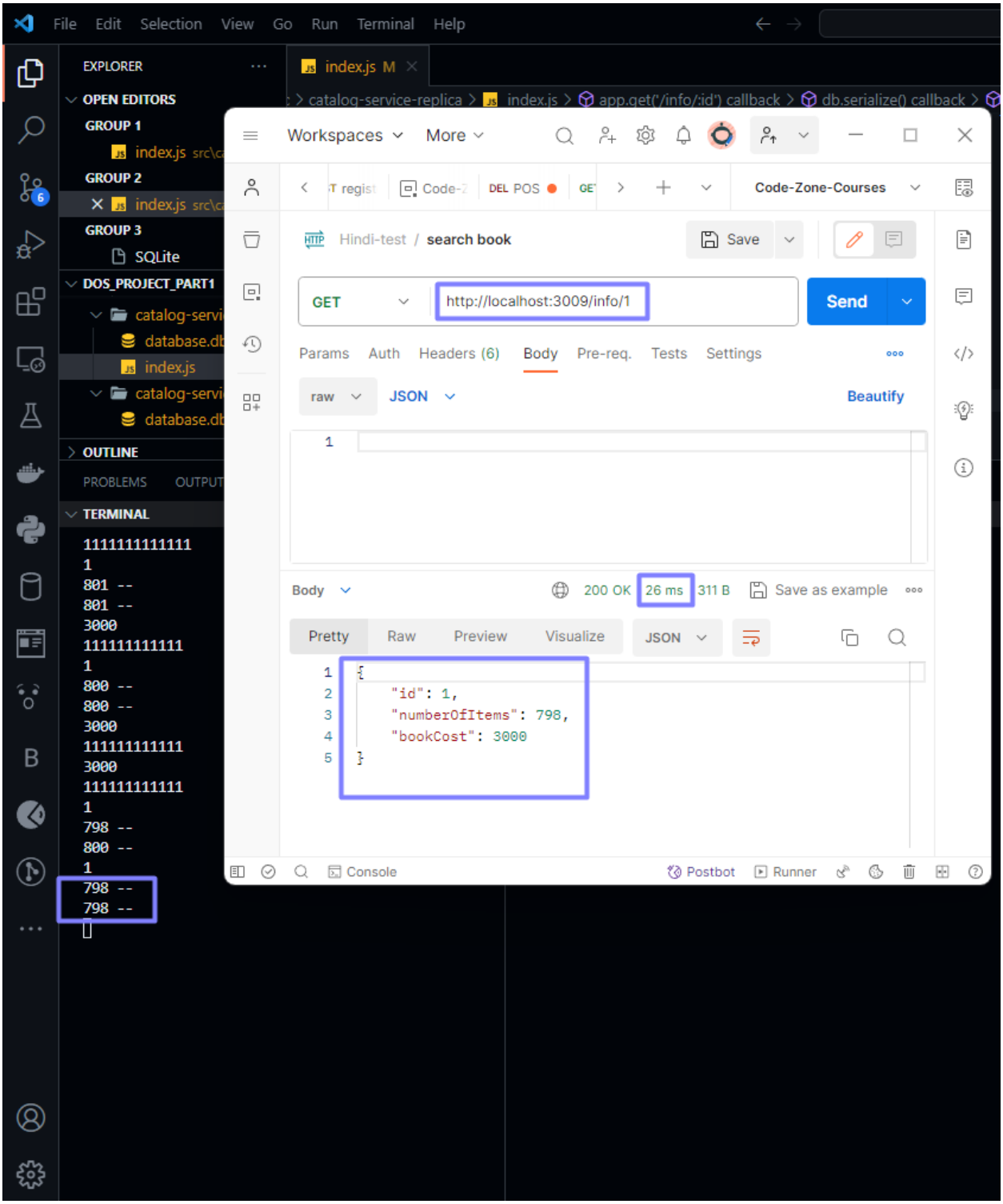
- for Origianl services:




Workspaces
More
Code-Zone-Courses
Hindi-test / search book
Save
GET
http://localhost:3005/info/1
Send
Params
Auth
Headers (6)
Body
Pre-req.
Tests
Settings
raw
JSON
Beautify
1
Body
200 OK
30 ms
311 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"numberOfItems": 800,
"bookCost": 3000
}
Console
Postbot
Runner
3000
1111111111111
1
801 --
802 --
1
801 --
801 --
3000
1111111111111
1
800 --
801 --
1
800 --
800 --

- for Replica Service

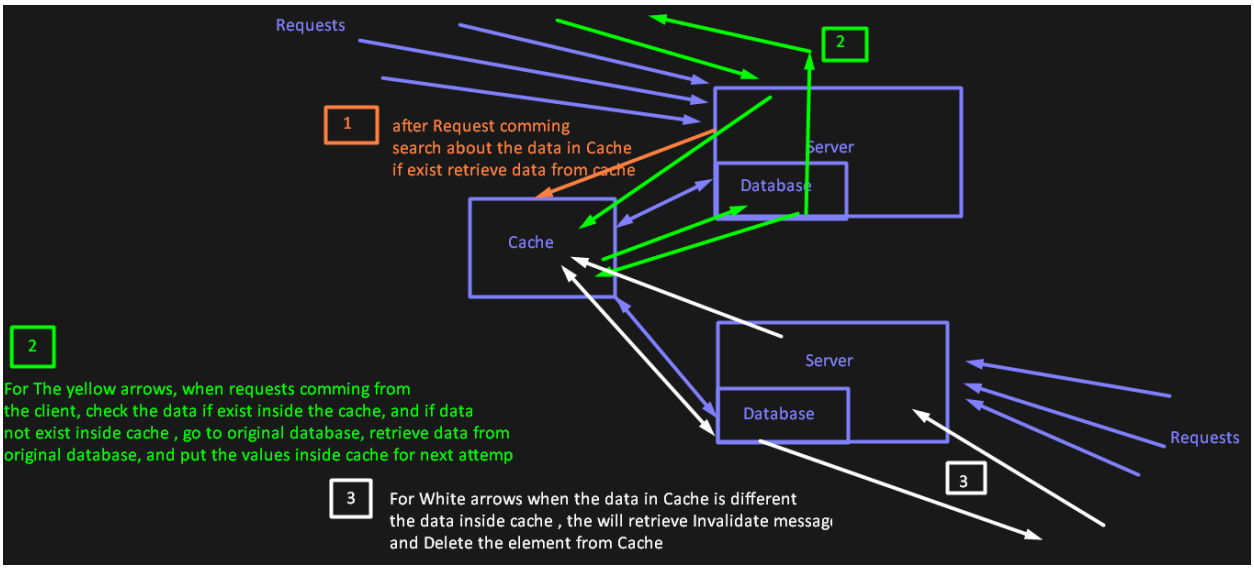




Expirement Run	Without Cache	With Cache	#of Times when using Cache speed

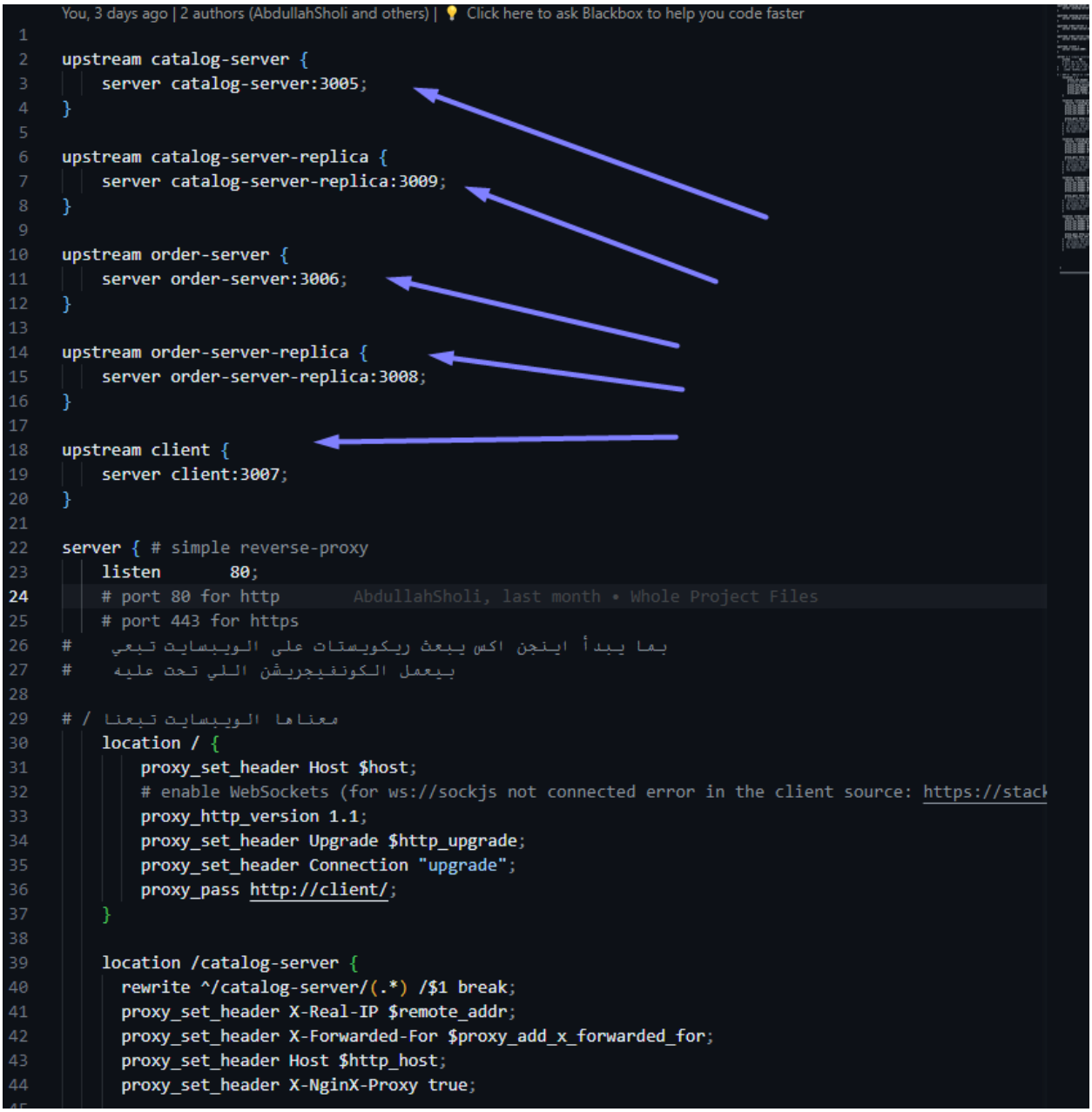
1	97	34	97/34 = 2.853 Times
2	29	17	29/17 = 1.71 Times
3	40	14	40/14 = 2.857 Times

The Procedure:

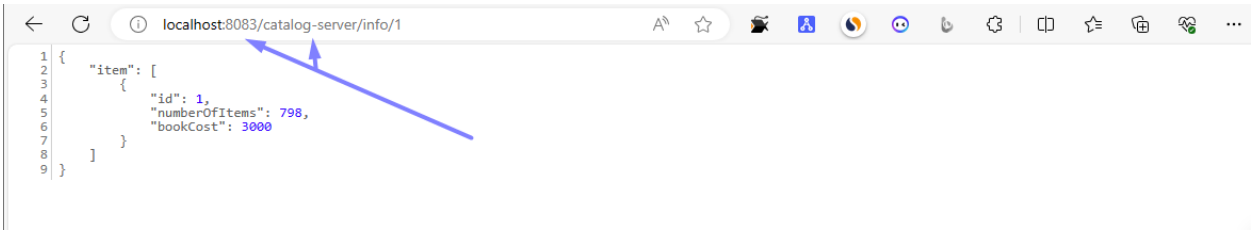


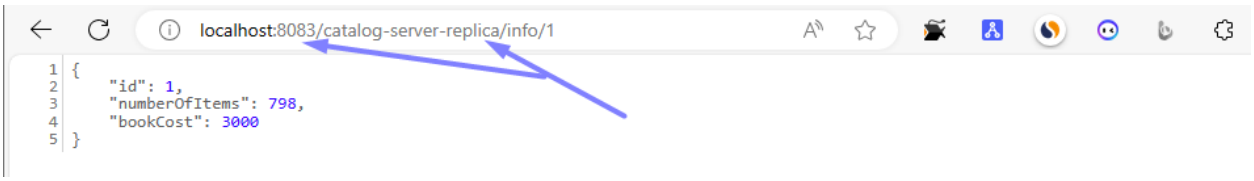
Part2: Loadbalance with NGINX

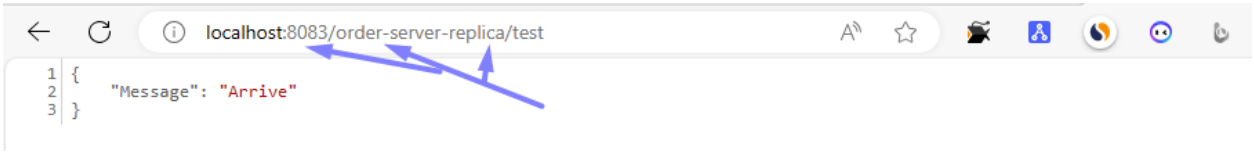
I Used Nginx to acheive loadbalance , each service exist in it's seperate Docker Container and it has own Inerface & Port to communicate with other services, Below my File Configuration for NGINX

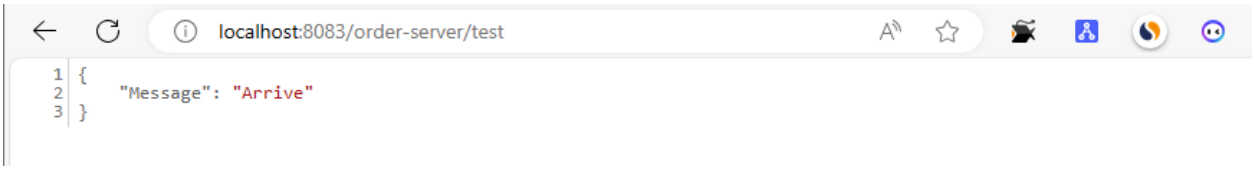


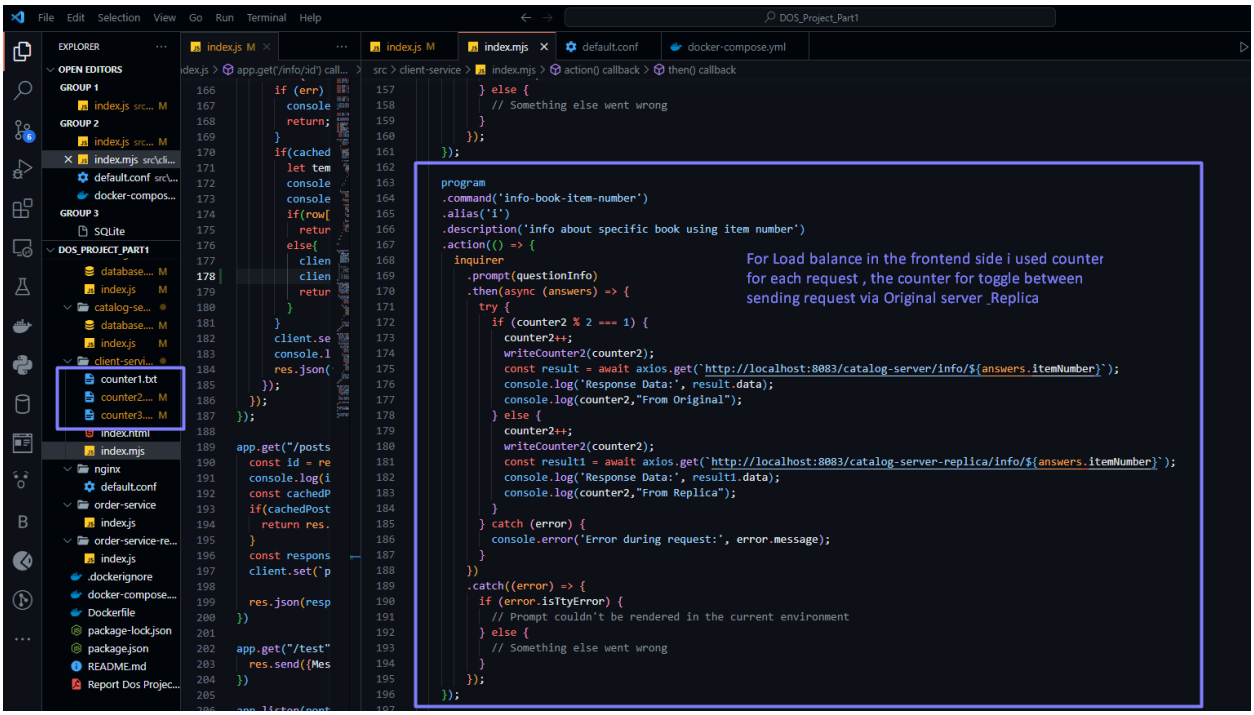
for testing all container running correctly via nginx:



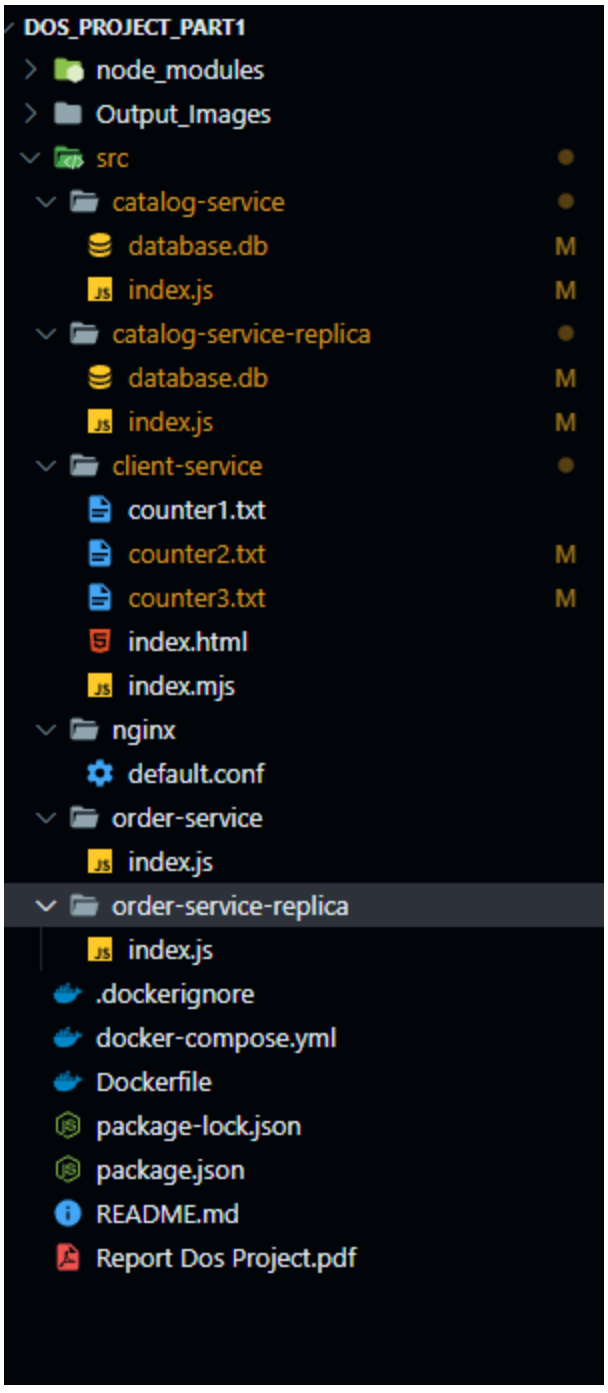








For Replication Services & Database:



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-r
      - ./src/nginx:/app/src/nginx:ro
      - ./src/catalog-service-replica/database.db:/app/database
    ports:
      - '3009:3009'
    environment:
      - PORT=3009
```

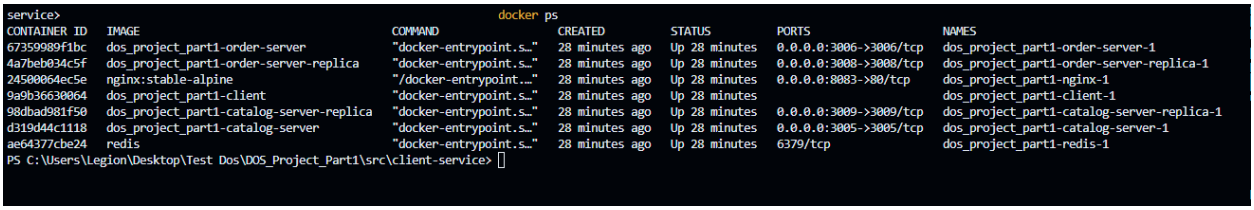
```
      - NODE_ENV=production2
  order-server:
    build:
      context: .
      target: production1
    volumes:
      - ./src/order-service:/app/src/order-service:ro
      - ./src/nginx:/app/src/nginx:ro
    ports:
      - '3006:3006'
    environment:
      - PORT=3006
      - NODE_ENV=production1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

  order-server-replica:
    build:
      context: .
      target: production3
    volumes:
      - ./src/order-service-replica:/app/src/order-service-repli
      - ./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:



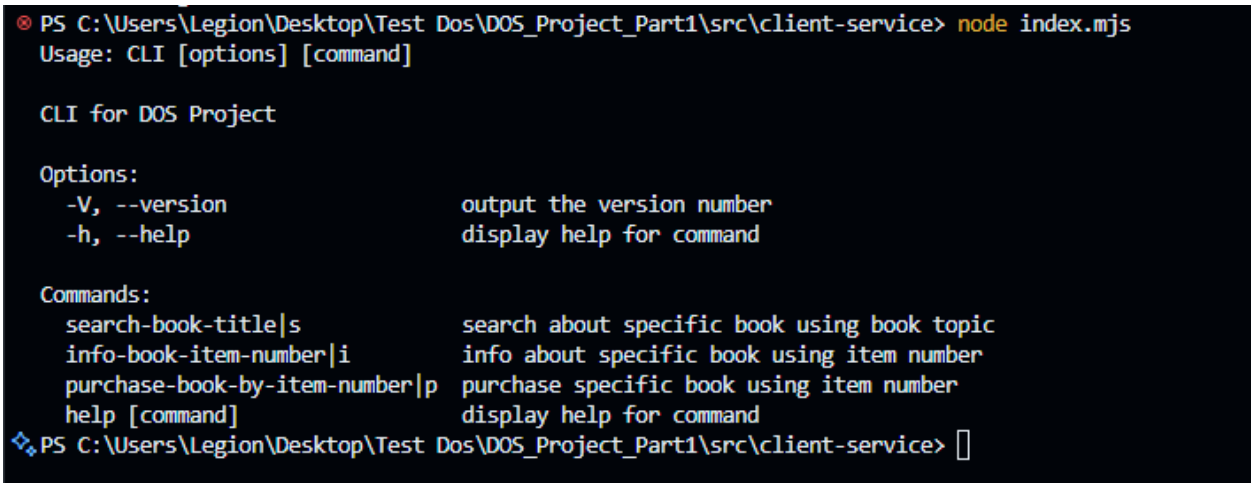
For Running The Project, it’s the same for Part1:

`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>