An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Distributed operating systems Project Report #1

An-Najah National University

Computer Engineering Department

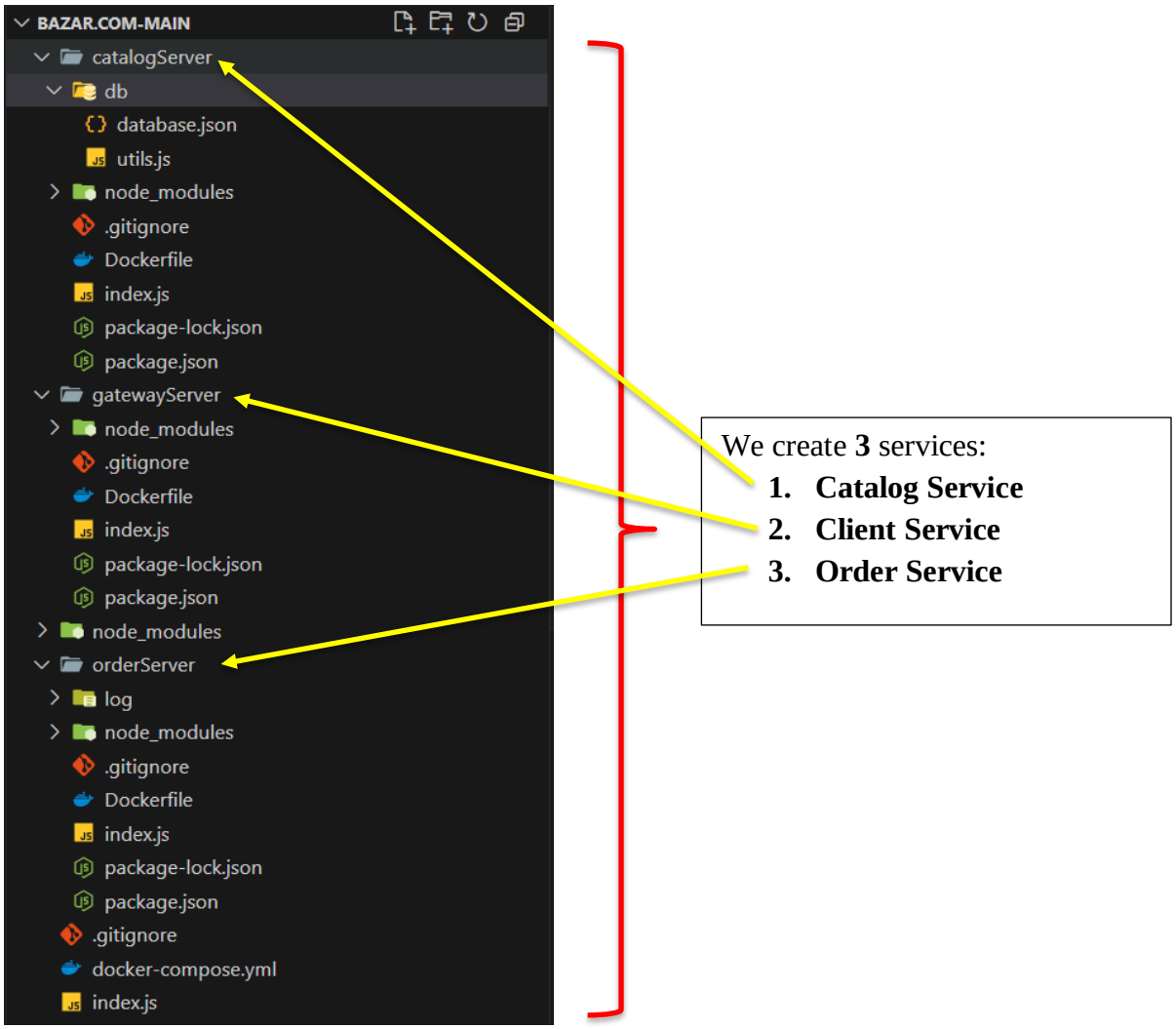
Students:

1. Ahmad Dweikat 12042774
2. Abd Alhameed Mizher 12029826

Submission Date: 4th April 2025

Project Structure:

- In first here our **project structure**



- We create **Dockerfile** to create our containers.
- We create container for each service by using **docker-compose.**
  - **Docker-compose:** is a tool that helps you run and manage multiple Docker containers easily.
  - We create **docker-compose.yml** to make the configuration for the project containers.

Code Explanation

Order Service

Dockerfile gatewayServer | Dockerfile orderServer

orderServer > Dockerfile > ...

```dockerfile
# Use an official Node.js runtime as the base image
FROM node:18-alpine

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Install nodemon globally
RUN npm install -g nodemon

# Copy the rest of the application code to the working directory
COPY . .

# Expose the port on which the app runs
EXPOSE 3002

# Command to run the application
CMD ["nodemon","--legacy-watch", "index.js"]
```

Catalog Service

Dockerfile gatewayServer | Dockerfile catalogServer

catalogServer > Dockerfile > ...

```dockerfile
# Use an official Node.js runtime as the base image
FROM node:18-alpine

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Install nodemon globally
RUN npm install -g nodemon

# Copy the rest of the application code to the working directory
COPY . .

# Expose the port on which the app runs
EXPOSE 3001

# Command to run the application
CMD ["nodemon","--legacy-watch", "index.js"]
```

Dockerfile

gatewayServer > Dockerfile > ...

```dockerfile
# Use an official Node.js runtime as the base image
FROM node:18-alpine

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json to the working directory
COPY package*.json ./

# Install dependencies
RUN npm install

# Install nodemon globally
RUN npm install -g nodemon

# Copy the rest of the application code to the working directory
COPY . .

# Expose the port on which the app runs
EXPOSE 3000

# Command to run the application
CMD ["nodemon","--legacy-watch", "index.js"]
```

We created 1 docker files for each service this one for Client Service

Copies package.json and package-lock.json to the container so we can install dependencies based on these files.

Runs npm install to install all the necessary packages defined in package.json.

Installs nodemon globally in the container

Copies all the remaining application

This tells Docker to expose port 3000, which is the port your app will run on inside the container.

This command runs the app with nodemon, enabling auto-restart on file changes.
The --legacy-watch flag ensures file watching works properly in Docker; index.js is the app's entry point.

docker-compose.yml

```yaml
version: "3.8"

Run All Services
services:
  Run Service
  gateway-server:
    build: ./gatewayServer
    ports:
      - "4000:4000"
    networks:
      - my_network
    volumes:
      - ./gatewayServer:/usr/src/app

  Run Service
  catalog-server:
    build: ./catalogServer
    ports:
      - "4001:4001"
    networks:
      - my_network
    volumes:
      - ./catalogServer:/usr/src/app

  Run Service
  order-server:
    build: ./orderServer
    ports:
      - "4002:4002"
    networks:
      - my_network
    volumes:
      - ./orderServer:/usr/src/app
```

gateway-server: Acts as the entry point, routing requests to other services via port **4000**.

catalog-server: Manages product/catalog data, accessible on port **4001**.

order-server: Handles order operations like placing or tracking, running on port **4002**.

Services Files:

1. **gateway_service: index.js**
2. **catalog_service: index.js**
3. **order_service: index.js**

before that here my **database.json** that contain books with details.

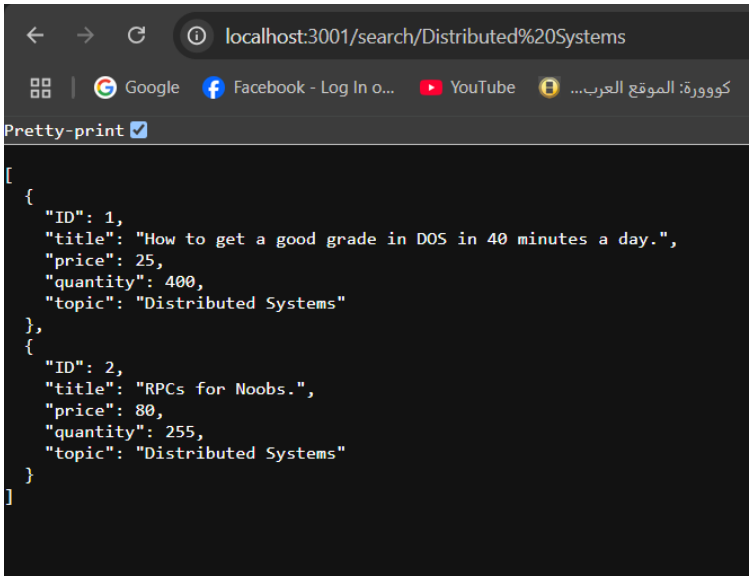
database.json

catalogServer > db > database.json > ...

```
[
  {
    "ID": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 30,
    "quantity": 210,
    "topic": "Distributed Systems"
  },
  {
    "ID": 2,
    "title": "RPCs for Noobs.",
    "price": 180,
    "quantity": 80,
    "topic": "Distributed Systems"
  },
  {
    "ID": 3,
    "title": "Xen and the Art of Surviving Undergraduate School.",
    "price": 330,
    "quantity": 250,
    "topic": "Undergraduate School"
  },
  {
    "ID": 4,
    "title": "Cooking for the Impatient Undergrad.",
    "price": 185,
    "quantity": 365,
    "topic": "Undergraduate School"
  }
]
```

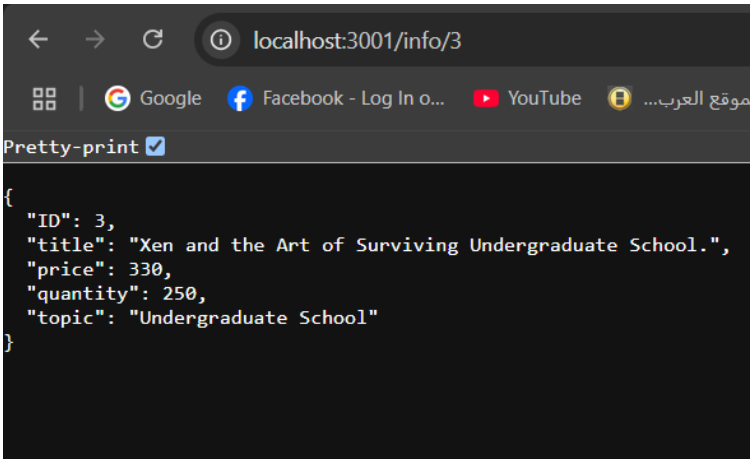
catalog_service (Requests):

- **Search by topic**
  - **localhost:3001/search/:bookTopic**
- **Example**
  - **localhost:3001/search/Distributed%20Systems**



Here after search for topic Distributed Systems

- **Search by ID**
  - **localhost:3001/info/:bookID**
- **Example**
  - **localhost:3001/info/3**



Here after search for ID: 3

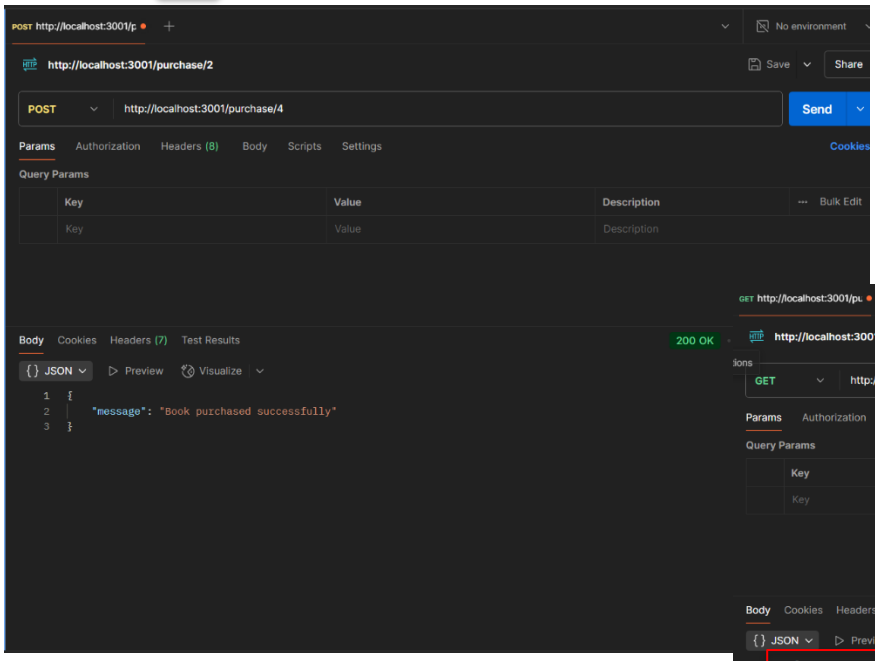
- **purchase a book**
  - **localhost:3001/purchase/:bookID**
- **Example**
  - **localhost:3001/purchase/4**

Before

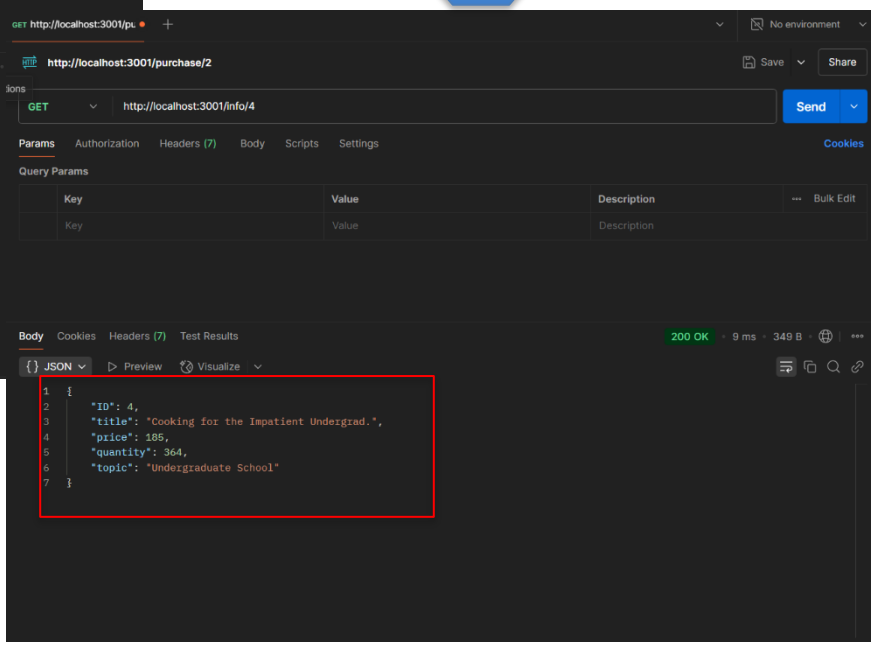
```
24      "ID": 4,
25      "title": "Cooking for the Impatient Undergrad.",
26      "price": 185,
27      "quantity": 365,
28      "topic": "Undergraduate School"
29    }
30  ]
```

After

1



2




```
1  {
2      "ID": 4,
3      "title": "Cooking for the Impatient Undergrad.",
4      "price": 185,
5      "quantity": 364,
6      "topic": "Undergraduate School"
7  }
```

Commands

docker-compose up -d --build → to build all containers
docker-compose down → to stop all containers

Run each node file using
node index.js after going to its directory (3 terminals : 1 for client , 1 for catalog and 1 for order).