Project

Fruit Price Microservices System

Background

This project is a continuation of your previous microservices exercise involving currency exchange. Now, instead of currency rates, you'll work with **fruit prices across months**, and apply the same principles of **microservices**, **service communication**, and **basic calculations**.

📦 System Overview

Your system must consist of:

🔹 1. A Database

- Contains fruit prices in USD per kilogram (or pound) for every month.
- Source: An Excel file provided to you, structured as follows:
  - **Rows** = Fruits (e.g., Banana, Apple, Cherry, etc.)
  - **Columns** = Months (January to December)
  - **Cells** = Price of 1kg for that fruit in that month

🔹 2. Microservice 1: Fruit Month Price Service (FMP)

- **Purpose**: Query the price of a fruit in a specific month.
- **Input**: `fruit` and `month` via URL.
- **Endpoint**: `GET /fruit-price/fruit/{fruitName}/month/{monthName}`
- **Sample Output (JSON)**:

```
{
  "fruit": "banana",
  "month": "july",
  "fmp": 8.83,
  "port": 8000
}
```

🔹 3. Microservice 2: Fruit Total Price Service (FTP)

- **Purpose**: Calculate the total cost of purchasing a quantity of a fruit in a specific month.
- **Input**: `fruit`, `month`, and `quantity` via URL.
- **Endpoint:**

```
GET /fruit-total/fruit/{fruitName}/month/{monthName}/quantity/{quantity}
```

- **Sample Output (JSON)**:

```
{
  "fruit": "banana",
  "month": "july",
  "fmp": 8.83,
  "quantity": 10,
  "total": 88.3,
  "port": 8100
}
```

📋 Requirements

1. **Data Import**:
   - Excel file must be converted and stored in a database.
   - Schema: `fruit`, `month`, `price` (1kg).
2. **Service Communication**:
   - FTP service must **internally call** FMP service to obtain price.
3. **Environment Configuration**:
   - Use distinct ports (e.g., 8000 for FMP, 8100 for FTP).
   - Optional: use **Spring Cloud Config**, **Eureka**, or **Docker**.
4. **RESTful Design**:
   - Use **RESTful endpoints**, clear naming, and **HTTP GET** requests.

5. **Demonstration Tools**:
    - You may use **any client tool** to demonstrate your services: browser, **Postman**, **curl**, or others.
6. **Presentation and Verification**:
    - A **PDF report is not required**. Instead, you must **orally explain your implementation during your live demo**. The TA will use this interaction to verify that the work is genuinely your own (not generated by others or AI).
7. **Code Submission**:
    - Submit either by uploading your **project files to Moodle**, or by pushing to GitHub and providing the GitHub link within your Moodle submission.
    - **Email submissions will not be accepted.**

📦 What to Submit

- Working system with two **Spring Boot microservices**
- Database populated from the Excel file
- Proper **REST endpoints** with correct **JSON responses**
- Clean, modular code structure
- GitHub link (optional, if applicable)
- **Live presentation to your TA** (submission + demo)

🧪 Live Demo Requirements (Mandatory and Short)

You must demonstrate:

- Both services running
- Functionality tested via browser, Postman, or curl, or etc.
- Code walkthrough
- Clear explanation of how FTP calls FMP internally
- Discussion of implementation choices

📝 Grading Rubric (100%)

Component	Points
Data import from Excel to DB	10%
Fruit Month Price Service (FMP)	20%
Fruit Total Price Service (FTP)	20%
Microservice Communication (FTP → FMP)	15%
Correct JSON Output	10%
RESTful Endpoint Design	10%
Port Separation & Optional Config Use	10%
Code Quality and Structure	5%
TA Presentation (Live Demo)	**MUST**