Grocery Microservice Application

Introduction:

This project is a microservices-based Grocery management system that allows users to: Register and authenticate, buy products, sell products, payment, get Invoice.

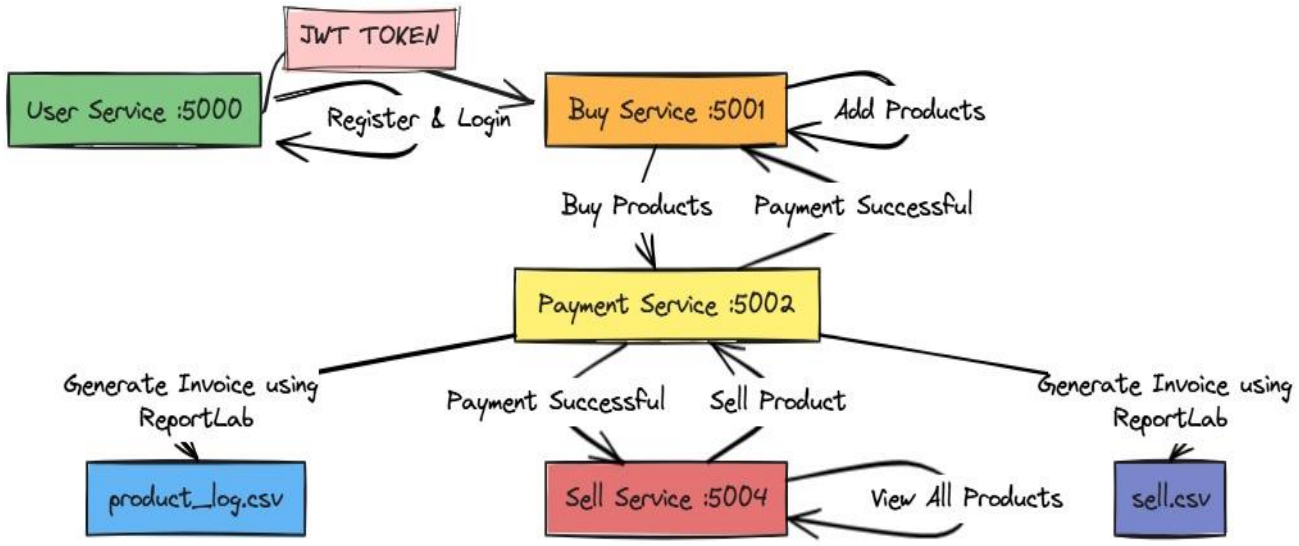
Technologies Used:

- Python-Flask = Backend Service
- PostgreSQL = Database
- JWT (JSON Web Tokens) = Authentication

Requirements:

- Flask==2.3.3
- Flask-JWT-Extended==4.5.2
- requests==2.31.0
- python-dotenv==1.0.0
- reportlab==4.0.7

Microservice Architecture Diagram



API Communication Flow:

User_Service:[port:5000]

- POST /register → Creates a new user.
- POST /login → Authenticates and returns a JWT token.

Buy_Service: [port:5001]

- POST/add_products →Adds product
- POST/buy_products →Buy Products

Sell_Service: [port:5003]

- GET/all_products→ View products
- POST/sell_products→Sell products

Payment_service: [port:5002]

- POST/payment→payment for purchased products

Sample JSON for Testing APIs:

1. User Registration

   POST /register

   ```
   {
     "name": "Ravi",
     "email": "ravi@example.com",
     "password": "Pass@123"
   }
   ```

2. User Login

   POST/login

   ```
   {
     "email": "ravi@example.com",
     "password": "Pass@123"
   }
   ```

3. Add products

POST /add_products

Headers:

Authorization: Bearer <JWT_TOKEN>

```
{
  "product_name": "Tata Salt",
  "quantity": 100,
  "amount": 20.5
}
```

4. Buy products

POST /buy_products

Headers:

Authorization: Bearer <JWT_TOKEN>

```
{
  "product_name": "Tata Salt",
  "quantity": 5
}
```

5. Payment

POST/payment

```
{
  "amount": 102.50,
  "user_id": "ravi@example.com",
  "product_name": "Basmati Rice",
  "quantity": 2,
  "price_per_unit": 51.25
}
```

6. Invoice generator:

Tool:ReportLab

Output:

invoice_data = {

"Invoice ID": "INV-000123",

"Transaction ID": "TXN-456789",

"User ID": 101,

"Product Name": "Wheat Floor",

"Quantity": 2,

"Price per Unit": "₹500",

"Total Cost": "₹1000",

"Payment Status": "Success",

"Sold At": datetime.utcnow().isoformat(),

}

CONCLUTION:

This Microservice Architecture Diagram illustrates a modular and scalable system design where each service performs a specific function and communicates through defined APIs. The User Service, Buy Service, Payment Service, and Sell Service operate independently, flexibility, and ease of maintenance.

REFERENCE:

https://dev.to/behalf/authentication-authorization-in-microservices-architecture-part-i-2cn0

Integrating Payment Gateways in Flask: A Developer's Guide

Mastering PDF Report Generation with ReportLab: A Comprehensive Tutorial Part 1 | by Praveen Goyal | Medium

https://l1nq.com/tZoMc