Sweet Home Application

List of Services:

1. Eureka-Server (PORT: 8761)
2. API-Gateway (PORT: 9191)
3. Booking-Service (PORT: 8081)
4. Payment-Service (PORT: 8083)

Deployment Sequence:

Eureka_Server → API_Gateway → Payment_Service → Booking_Service

Application Logic:

- Booking, Payment and API-Gateway services are registered in Eureka client for ease of discovery and communication; so, Eureka server is deployed first**.**
- The API-gateway (hosted on port – 9191) acts as the entry point for all the services in the application, and hence requests to all services have the common domain name of localhost://9191; hence API-Gateway is deployed second.
- The Payment service contains **1 POST method** for creating payment transaction, and **1 GET method** for reading payment transaction
    - The postPaymentTransaction method receives the payment info from the postPaymentInfo method of booking service, and saves the transaction in payment-service database; hence payment-service must be deployed third.
    - The getPaymentTransaction method is an internal call which reads the transaction stored in payment service based on ID.
- The booking service contains **2 post methods**, one for **posting booking info** and other for **posting payment info**.
    - The postBookingInfo method internally saves booking related information within booking service database.
    - The postPaymentInfo validates ID and payment method inputs provided and then sends payment info details to Payment Service for creating a transaction and saving in payment database; hence booking-service is dependent on payment-service and is deployed last.

Configuration Settings:

- Each service has application. Properties / application.yml files where the configuration is done. Config-server is not used for simplicity as it is not a requirement.

How to Test calls:

- Since API gateway is implemented with predicates and context paths as configured in POSTMAN API documentation, the end points can be tested as is.
- Some calls such as the **POST** method: postingBookingInfo in booking service, and the **GET** method: getPaymentTransaction method, will throw run-time exceptions and Internal server error with status code 500 on postman with messages printed on console if invalid inputs are provided such as numberOfRooms < 0, or getting payment transaction for ID that does not exist.