MicroServices Case Study

This document is prepared to define a use case for the candidates who are appearing for MicroServices training online.

# Instructions

1. The trainer of Udemy MicroServices Course has used Open source tool and other technologies for demos including MacOS.
2. Candidate **is not** supposed to have the same Machine, Cloud and OS setup on their Machine as Udemy Trainer has.
3. Candidate should be able to develop MicroServices on their own Machine and respective OS.
4. If there is any infrastructure setup issue or challenge, Candidate can connect with the concerned SPOC to clarify the doubts.
5. Candidates can take help outside of the Udemy Video course contents to complete this Case study.
6. Each Candidate will be developing 4 Spring Boot based MicroServices apart from supporting services e.g. API Gateway, Service Discovery etc.

# Pre-requisites

1. Candidate has completed his/her video training on Udemy for MicroServices course.
2. Ensure that following tools are installed on your machine:
   1. JDK 7 or 8
   2. Spring Tool Suite (STS) or any other IDE whichever they are comfortable with.
   3. Maven

# Case Study

Please ensure that all the instructions are followed before you start to work on this Case Study

An organization ***XYZ Technologies Ltd.*** has a requirement to develop **Order Management System** using MicroServices Architecture. You are asked to develop 4 different independent services named *User-Service, Product-Service, Cart-Service,* and *Order-Service* which should run independently while communicating with each other.

* User service should have following operations:
  + Login User
* Product Service has following operations:
  + Get all products
  + Get a Product based on product id
* Cart Service has following operations:
  + Allow user to add products to the cart (Integration with Product Service to fetch Products)
  + Remove products from cart
* Order Service has following operations:
  + User can place an order (products should be deleted from cart after placing order – Integration with Product Service)
* Micro Service Configurations:
  + Each MicroService should be registered to Service Registry & Discovery service (Eureka)
  + Each MicroService should use API Gateway service (Zuul)
  + Each service should be fault tolerant (Hystrix) and show the operation execution on Turbine Dashboard

# Assumptions

1. You can use either memory or database for data storage.
2. You can use/declare member variables, table relationships as per your understanding.
3. You can use any port for any service, given that port is not occupied/used by any other application.

# Evaluation parameter

Each candidate will be evaluated based on following parameters:

1. No need of **User Interface** development. The services can be tested using POSTMAN
2. Candidate should have 4 independent working services (*User-Service, Product-Service, Cart-Service,* and *Order-Service*) and 4 supporting services (*Eureka-Server, Zuul-Service*).
3. All the operations mentioned should be working properly and providing expected output.
4. REST endpoints should be resource name (nouns) with proper naming convention, not verbs.
5. Appropriate HTTP methods are used on resources for operations.
6. All services should be registered to Service Discovery (Eureka), go to Eureka web interface.
7. All service communication should be happening via API Gateway (Zuul).
8. All operations should be visible on Circuit Breaker dashboard (Hystrix & Turbine), go to Hystrix dashboard.
9. Distributed log tracing should be visible on Zipkin dashboard.