# Micro Services Project with Spring Boot

## 1. Instrucitons for start up

Please execute the deploy.sh file inside each folder in the following order

1. Eureka Server
2. API Gateway
3. Booking Service
4. Payment Service

- Please allow some time for the applicaitons to register on the eureka server and api gateway before testing the api end points.

---

## 2. General Information

The Booking and payment services use H2 Embedded Database. They can be accessed using the following credentials:

1. Booking service
   - address: localhost:8081/h2-console
   - username - bs
   - JDBC url - jdbc:h2:mem:bookingdb
   - password - ""
2. Payment service
   - address: localhost:8083/h2-console
   - username - ps
   - JDBC url - jdbc:h2:mem:paymentdb
   - password - ""

- All API end points are as per requirents

---

## 3. Special Notes

- Datatype used for fromDate and toDate is LocalDate as per database schema. Same is apt for real world scenarios.
- Datatype for bookedOn is LocalDateTime as apt for real world scenarios

---

## 4. Coding Logic

Both the services use the Three Tier Architecture Pattern. An error controller is also implemented to handle erros in a very basic manner. Hitting any url inside the services will trigger the error handler.

### 2.1 Booking Service

The service is implemented as per requirements and API end points have been tested with the given API Documentation

- The create booking function takes in the required details, calculates the pricing as per the number of rooms required, generates the list of room numbers and also updates the bookedOn time internally in the service layer.
- The second API (for making payment for created bookings) uses **RestTemplate** to make calls to the **Payment Service**. The transaction id received is then set using setter to the BookingInfoEntity retrieved from the database using the id passed to the function as parameter. The updated entity is then used to update the database internally.
- Confirmation of booking is printed on the console as per requirements

### 2.2 Payment Service

- This service is implemented as a very basic spring boot application that fulfills the requirements.
- Additionally, http://localhost:9191/payment/transaction/{id} is also designed to handle invalid transaction id inputs appropriately.

---

## Summary

Developing these applicaitons has been a good learning experience. It has helped me understand some of the tricky problems one might encounter while developing microservices and fix them to create a working applicaiton.

Thanks to upgrad for giving us the opportunity to work on a project such as this one.

– Daniel William