HCG

INTERVIEW TEST
BACK-END ENGINEER

Hash Consulting Grp Pte. Ltd.
3 Fraser Street, #8-21 DUO Tower, Singapore 189352
info@the-hash.com | www.the-hash.com

HCG

Table of Contents

Interview Test

Position: Back-end Engineer

I. Technical Requirements

- Develop back-end services using Spring Boot and Maven.
- Java 1.8 or 17 will be used as the development language.
- MySQL or MariaDB will be used as the relational database management system.
- RESTful APIs will be the primary data exchange method, with GraphQL being preferred for public APIs but not required.
- Unit tests will be written using JUnit.
- Public API documentation will be generated using Swagger.

II. Instructions for Submission

- Send the public repositories and related resources via email: **careers@the-hash.com**
- Completing all tasks is optional but preferred.
- The maximum score achievable is **100 points**, and in order to pass the exam, a minimum of **70 points** is required.
- Duration: **2 days.**

III. Test Contents

1. Key Objectives

- Design and implement APIs for a simple hotel booking system using Java.
- As a back-end engineer, the interviewee will need to design and develop back-end services.
- There are 6 objectives to achieve the goal. Each objective will include one or many tasks.
- The interviewee will gain a score after completing a task. Maximum score is 100 points.

2. Objective 1: Setup **(5 points)**

- Set up 3 project repositories on GitHub or GitLab following the Technical Requirements section and below structure: **(3 points)**

  - The primary repository is **hcg-interview-services-public**. This repository contains only public APIs published for public using such as integration with front-end applications.
  - The second repository is **hcg-interview-services-booking**. This service will handle all business requirement relating to booking.
  - The last repository is **hcg-interview-services-inventory**. Inventory data such as room types, rate plans, room rates, etc. will be handled in this service.
- Draw a overview diagram for indicating the relationship between these services and between the back-end services and front-end applications (how front-end applications communicate with back-end services). **(2 points)**

3. Objective 2: Database Design (25 points)

- Design the database with below requirements:
  - Hotel Information.
  - Room type: room type is the type of one or many hotel rooms. Some popular room types: Single Room, Double Room, Family Room...
    - Each hotel includes one or many room types.
    - Each room type is mapped to one or many rate plans.
  - Rate Plan: rate plan is a plan for adjusting room rate for one or many period of time. For example, hotel owner creates a rate plan for adjusting room rate on summer, with higher room rate and extra promotions if the guest comes with his/her family members.
    - Each hotel includes one or many rate plans.
    - Each rate plan is mapped to one or many rooms.
    - There is always a default rate plan for standard room rate.
  - Room rate: room rate is the price for selling room in a day.
    - Room rate may be different for each date.
    - Room rate is unique for each combination of room type, rate plan and date.
  - Room availability: number of rooms can be sold for each room type in a day.
    - Room availability may be different for each date.

        - Room availability is unique for each combination of room type and date.
    - Booking:
        - Each booking is mapped to one room type and one rate plan.
        - Each booking is mapped to one or many guests.
        - Each booking has a unique booking number.
        - Price of booking must be the value at the time of booking.
    - Guest.
        - Guest is mapped to one or many bookings.
        - Guest information must contain enough information for hotel to contact the guest.
- Regarding this objective, the interviewee need providing the following resources:
    - Entity Relationship Diagram (ERD) **(10 points)**
    - SQL scripts to generate the database and tables. **(10 points)**
    - SQL scripts to generate mock data for tables. **(5 points)**

4. Objective 3: API Development (35 points)

For a simple hotel booking system, the interviewee needs to create APIs to handle the four below functions:

- Get the room rate and room availability for each room type and rate plan in a period of time. **(10 points)**
    - For example, the guest want to view the hotel availability from 1st January to 1st February, the API will return the room rate and room availability for each combination of room type, rate plan and date from 1st January to 1st February.

- Sample response

```
{
  "data": [
    {
      "date": "202x-01-01",
      "roomAvailability": [
        {
          "roomTypeId": "1",
          "availableToSell": 1
        },
        {
          "roomTypeId": "2",
          "availableToSell": 3
        },
        ...
        {
          "roomTypeId": "99",
          "availableToSell": 0
        }
      ],
      "roomRate": [
        {
          "roomTypeId": "1",
          "ratePlanId": "R1",
          "price": 100.00,
          "currency": "USD"
        },
        {
          "roomTypeId": "1",
          "ratePlanId": "R2",
          "price": 150.00,
          "currency": "USD"
        },
        {
          "roomTypeId": "2",
          "ratePlanId": "R1",
          "price": 200.00,
          "currency": "USD"
        },
        ...
        {
          "roomTypeId": "99",
          "ratePlanId": "R99",
          "price": 100.00,
          "currency": "USD"
        }
      ]
    },
    ...
  ]
}
```

- Create a new booking. **(10 points)**
    - The selected room type must be available in the staying nights of booking. The availability is validated from arrival date to the day before departure date.
    - The total price is calculated from arrival date to the day before departure date.
    - Booking has one guest at least.
    - The guest name and email address must be validated.
    - After creating the booking successfully, the room availability needs to be updated.
- Change the arrival and departure date of a booking. **(10 points)**
    - The booking must be existed.
    - The new staying period must be available.
    - After modifying the booking successfully, the room availability needs to be updated.
- Cancel a booking. **(5 points)**
    - The booking must be existed.
    - After cancelling the booking successfully, the room availability needs to be updated.

5. **Objective 4: Unit Testing (10 points)**

   Write unit tests for implemented APIs from Objective 3. **(10 points)**

6. **Objective 5: Documentation (10 points)**

   Use Swagger to generate API document for above APIs defined in Objective 3. **(10 points)**

   Note: to acquire full score, the API document must follow the conventions of RESTful API definition and have description for API operations and every field of model.

7. **Objective 6: Enhancement (15 points)**
   - Implement Dockerfile for each repository. **(5 points)**
   - Setup a message broker using Apache ActiveMQ. **(10 points)**
       - Booking service is the message producer.
       - Inventory service is the message listener.

- When a booking is created/modified/cancelled, send a message to Inventory service to update the room availability instead of sending the requests directly through APIs.