Hotel Room Reservation System

Contents

[1.0 Problem statement 2](#_Toc46520273)

[2.0 Skills to develop the project 2](#_Toc46520274)

[3.0 Use Case Diagram for the Problem Statement 3](#_Toc46520275)

[4.0 User Stories 3](#_Toc46520276)

[5.0 Expected Deliverables 4](#_Toc46520277)

[6.0 Milestone and duration 4](#_Toc46520278)

[7.0 Implementation Notes 5](#_Toc46520279)

[8.0 Evaluation rubrics 5](#_Toc46520280)

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Hotel Room Reservation System**” to be developed. The application should be built on Cloud Native Architecture with Microservice. Both functional and non-functional requirements are captured in this document.

**About the System**

The client would like to develop an independent application ‘Hotel Room Reservation system’ application in order to search for room availability, book room, view reservation and cancel reservation.

* All the functionality will be served as a Service via REST end point.

The following section will cover aspects related to Application.

1. Room Search
2. Room Booking
3. View Reservation
4. Cancel Reservation

**Scope of the System**

The scope of the system is explained through its modules as follows

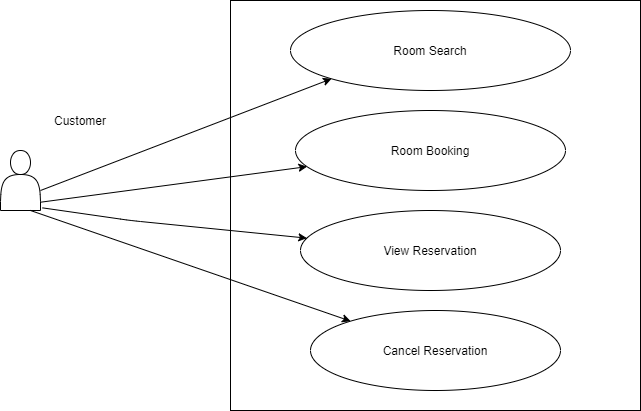
* Room Search – used by customers to search rooms in a hotel for provided date range. Microservice 1. .Use OKHttpClient library to call search service from microservice 3
* Room Booking - will be used by customers to book rooms in a Hotel.(Kafka EMQ) Microservice 2
* View Reservation - will be used by customers to view a Reservation details. Microservice 3
* Cancel reservation – will be used by customers to cancel Reservation. Microservice 3
* Also In microservice 3, write search api. Call this api from microservice 1 Skills to develop the project

Associate will implement skills from Backend and AWS Cloud platform to develop the application.

Below are the skill details.

|  |  |
| --- | --- |
| **Tower Name** | **Topics** |
| Backed - Java | Data Structures  Core Java 8  Spring Boot, JSON  Spring Cloud  Microservices  Junit  Data Structures |
| Cloud - AWS | Amazon Beanstalk, Amazon ECS, Amazon RDS,Amazon CloudWatch, AWS Code Commit, Code pipeline |

# Use Case Diagram for the Problem Statement

****

# User Stories

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | Room Search | Customers can search rooms in a hotel for provided date range and room type (optional).  Acceptance criteria:  It will return the available rooms for the search criteria as a json structure with proper http status code |
| US\_02 | Room Booking | Customer will book a room with hotel, date, room type choice, customer details with no of room’s information.  Acceptance criteria:  It will return the status of the booking either success or failure with reason with proper http status code. |
| US\_03 | View Reservation | Customer can view a reservation details by providing reservation number  Acceptance criteria:  It will return all the reservation details of that reservation if it is valid or return proper error message with http status code |
| US\_04 | Cancel Reservation | Customer can cancel reservation by providing reservation details.  Acceptance criteria:  It will cancel the reservation and free the booked room for future booking. |

# Expected Deliverables

The following deliverables are expected as outcomes

* Application Code base
* API for each service
* Junit coverage
* Readme document on the complete application

# Milestone and duration

As per project requirement, modification can be done in the below table.

|  |  |
| --- | --- |
| Milestone | Topic |
| Milestone -1 | Design and develop micro service application using backend technology and running the application in local system. |
| Milestone-2 | Use AWS component to build and deploy the application. Use cloud provider’s DevOps tools to setup a CI/CD pipeline to automate your releases. |

# Implementation Notes

|  |  |
| --- | --- |
| Backend – Java | Milestone-1   * Use Rest APIs to develop the services * Use Microservice Architecture * Use Domain Driven Design * Use AWS RDS to store / view data (local MySQL and the migration to RDS) * Use Swagger definitions * POST Man to invoke APIs * Any error message or exception should be logged (and help in refactor) * Unit test the application (Mockito) >80% * Raising Pull Requests, closing them are highly encouraged * All implementation should follow coding standards, secure coding practices, good code coverage. * Message input/output format should be in JSON (Read the values from the property/input files, wherever applicable). Input/output format can be designed as per the discretion of the participant * Web service URLs should be configurable.(No need for this) * Implement External Configurable Solution(Application .properties, config server) * Implement 12 Factor principles ( Follow the principals) |
| Cloud – AWS (No Need at this point) | Milestone-2   * Use AWS component to build and deploy the application. * Use cloud provider’s DevOps tools to setup a CI/CD pipeline to automate your releases. |

# Evaluation rubrics

|  |  |
| --- | --- |
| Microservices | * Follow the below basic structure   + **API - Controllers**   + **Domain - Model, Events, Business Services Integration**   + **Services – API Implementation**   + Infrastructure Project * Associate must have designed/developed Microservices as per the requirement * Each of the Microservices need to comprise below functionality, which need to be developed * Entity & Model classes, including appropriate relationship (like One-One, Many-One, etc…) between Entity Classes. (Entity and Model classes have been developed in the Previous Phase) * In case specific Entity or Model classes are required across multiple Microservices, it is recommended to maintain separate copy of Entity or Model classes for each Microservices. * Microservices should interact with corresponding databases it owns. * Microservice need to interact with other Microservice * Usage of Postman to test the Microservices by directly passing requests to each REST end Point, of each Microservice * Circuit Breaker, Service Registry, Service Discovery should be implemented |
| Rest API | * Associate must have used REST API for exposing resources * Associate must have used HTTP GET/PUT/POST request method designators for the business methods which is to be exposed * Associate must have customized the request and response formats according to the requirement * Associates must have used appropriate RETURN CODES based on the service outcome * Associates must have extracted query/form/header parameters from the input * Associate must have built a custom response based on the input * Use Swagger UI and test each public method in the service * Implementation of Repository pattern * Implement Service Registration and Discovery |
| Java | Associate should have used java 8 features, Control Statements and Operators, I/O Operations for implementing the functionalities. |
| Unit Testing | * Test cases covers the functionality of API with custom inputs * Good test Coverage |
| Common | * Code Smell * Technical Debt * Secured Coding * Coding Standards |
| AWS | * DevOps pipeline for microservices which uses cloud PaaS services to trigger a CI/CD pipeline when code is checked-in to GIT * The check-in process should trigger unit tests with mocked dependencies * DevOps dashboard should show status of CI/CD pipeline |