Hotel Reservation

Contents

[1.0 Problem statement 2](#_Toc112406256)

[2.0 Architecture Diagram for the Problem Statement 2](#_Toc112406257)

[3.0 Database Diagram for the Problem Statement 3](#_Toc112406258)

[4.0 Technical Scope 3](#_Toc112406259)

[5.0 Use case details 4](#_Toc112406260)

[6.0 Functional/Non-Functional Requirement of the Problem Statement 5](#_Toc112406261)

[7.0 Skills to develop the project 9](#_Toc112406262)

[8.0 Implementation Notes 10](#_Toc112406263)

[9.0 Evaluation rubrics 11](#_Toc112406264)

# Problem statement

The purpose of the requirements document is to systematically capture requirements for the project and the system “**Hotel Reservation**” to be developed. The application should be cloud native architecture with microservices. Both functional and non-functional requirements are captured in this document. It also serves as the input for the scope of the project.

**About the System**

The client would like to develop an independent application, **Hotel Reservation** to allow the customer to register themselves and find a room in the hotel. The portal provides facilities like customer creation, room creation and room reservation.

**Scope of the System**

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

* Customer Registration – used by the customer to create a new customer in the system
* Reservation – used by the customer to reserve a particular room on a particular date or delete the reservation if needed

# Architecture Diagram for the Problem Statement

**US\_01 Customer Registration**

Web Portal – Register Customer

Database

Validate Request and register customer in database

**US\_02 Login**

Database

Validate Request and generate token and login

Web Portal – Login

**US\_03 Room Reservation**

Database

Validate Request and create reservation in database

Web Portal – Create a Room Reservation

**US\_04 Delete Reservation**

Database

Validate Request and delete reservation from database

Web Portal – Delete an existing reservation

# Database Diagram for the Problem Statement

**Room**

1. roomId (int, id)
2. roomNbr(string)
3. roomType(string)

**Customer**

1. customerId (int) (id)
2. emailAddress (string)
3. password (string) (hashed)
4. firstName (string)
5. lastName (string)email
6. phoneNumber(string)
7. address(string)

**Reservation**

1. reservationId (int, id)
2. customerId (int)(fk)
3. roomId (int)(fk)
4. fromDate (date)
5. toDate (date)
6. checkInDone (Boolean)

# Technical Scope

|  |  |
| --- | --- |
| Compute & Integration | As an application developer, develop the application as a microservice architecture. And implement as follows:   1. Follow the Single Data Store per microservice practice 2. Document REST endpoints with OpenAPI or Swagger |
| Security & Identity | As an application developer:   1. Restrict the access over all write operation (secured operations) by adding authentication |
| Database & Storage | As an application developer:   1. Restructure the same as per microservice architecture 2. Use Cloud Spanner to storing the information |
| Governance & tooling | As an application developer:   1. Containerize the complete application 2. Perform unit testing of your application and do proper CI/CD |
| Code Quality/Optimizations | 1. Use SonarQube to scan the backend application for security vulnerabilities 2. Should have written clean code that is readable 3. Should have written testable code |

# Use case details

|  |  |  |
| --- | --- | --- |
| **User Story #** | **User Story Name** | **User Story** |
| US\_01 | User Registration | As a user, I should be able to register my details in the system.  Acceptance criteria:   * User should be able register the details in the system, and it should be saved in the database * Capture the details like First Name, Last Name, Address, Phone Number Email Address and Password * Email address should be User ID * Malicious text should be validated to prevent from CSRF and XSS attacks |
| US\_02 | Login | As a user, I should be able to login the system with valid credentials.  Acceptance criteria:   * User should be able login with their registered User Id aka Email Id and Password * When a user tries to login with incorrect credentials, valid error message should be shown |
| US\_03 | Creating Registration | As a user, I should be able to register a room at a particular date based on my preference  Acceptance criteria:   * User should be able to able to view available rooms as per their preference (roomType, date) * The search should check for existing reservations on that particular date before returning results * If no matching rooms found, the user should be told to change their date or preferences * If a valid room is found, the user should be able to reserve it. |
| US\_04 | Deleting a resevation | As a user, I should be able to delete an existing reservation  Acceptance criteria:   * User should be able to view their reservations on login, if needed they should be able to delete their reservation |

# Functional/Non-Functional Requirement of the Problem Statement

|  |  |
| --- | --- |
| US\_01 | User Registration |
| Description  Users should be able to register with valid details | |
| Input Parameters   * First Name * Last Name * Email ID * Password * Address * PhoneNumber | |
| Business Rules & Validations   * User Email ID should be considered as User ID. System should show success message with User ID * The user's first and last name should contain only letters and spaces * All fields should be mandatory * Email ID should contain “@” and”.” symbols and should be in valid Email format * Validation error messages should be displayed near the respective label * Email ID should be unique and should not be used previously for registration purposes * Password should contain at least one lower case letter, one upper case letter, one number, one special character and total length should be 10 * The password should be stored as an encrypted text in database | |

|  |  |
| --- | --- |
| US\_02 | User Login |
| Description  Users should be able to login with valid details | |
| Input Parameters   * User ID * Password | |
| Business Rules & Validations   * Shouldn’t allow submitting with empty password or empty User ID * Should show valid error message | |

|  |  |
| --- | --- |
| US\_03 | Creating a reservation |
| Description  The logged-in user should be able to find a room and create a reservation | |
| Find Room  Input Parameters   * RoomType, * FromDate * ToDate   Reserve  Input Parameters   * RoomId, * FromDate * ToDate | |
| Business Rules & Validations  Find Room Service   * The find room service should only unreserved room to the user.   **Reserve Room Service**   * All fields are mandatory * The customerId should be fetched for the logged in user * RoomId in token should be valid and not null * ToDate should not be before FromDate | |

|  |  |
| --- | --- |
| US\_04 | Deleting a Reservation |
| Description  The logged-in user should be able to delete his reservation | |
| Input Parameters   * ReservationId | |
| Business Rules & Validations   * All fields are mandatory * The user should be able to only delete his own reservation and not other’s reservation | |

**Service Requirements**

**US\_01 User Registration**

Once the user enters the details, they should be sent to the POST method and saved in the db. Mandatory fields should be validated as mentioned in the rules above and 400 status code exception, response should be sent with the missing field details. When the details are saved successfully, the service should response 200 ok along with success message. If there are any exceptions while connecting/saving to DB, the service should throw corresponding error with error status as 500.

**GET /register**

Header: NA

Body:

* FirstName
* LastName
* EmailID
* Password
* Address
* PhoneNumber

**US\_02 User Login**

Once the user enters the details, they should be sent to the POST method. Mandatory fields should be validated as mentioned in the rules above and 400 status code exception, response should be sent with the missing field details. When the details are authenticated successfully, the service should response 200 ok along with success message. If there are any exceptions while connecting/saving to DB, the service should throw corresponding error with error status as 500.

**POST /login**

Header: NA

Body:

* UserID
* Password

**US\_03 Create Reservation**

User should first put in his preferences and be able to fetch the available rooms

**GET /rooms/search**

Header:

* Token

Body:

* roomType
* fromDate
* toDate

**POST */reservation***

Header:

* Token

Body:

* roomId
* fromDate
* toDate

User should be able to reserve a room, buy passing necessary information

If the user id is not valid or the provided token is expired, then 401 will be returned. Request will be validated if model validation fails then service will return 400 as response. If the request is valid, then the question will be persisted to database and return 200 as response.

**US\_04 Deleting a reservation**

User should be able to see his reservation on the home page after logging in. The below API will be created to fetch existing reservations

**GET /reservations**

Header:

* Token

User should be able delete the existing reservation by calling the below method

**DELETE /reservations**

Header:

* Token

Body:

* reservationId

**Expected Deliverables**

The following deliverables are expected as outcomes:

* Application Code base
* Readme document on the complete application
  + Setup of the application
  + How to run the application
  + Any inference
  + Screenshot of UI results
* Reports:
  + Unit/Functional test report

# Skills to develop the project

List the Technology based on your respective technology stack, that will be used to develop the project.

|  |  |
| --- | --- |
| Skill Stack | Core Java |
| Front end | Angular,  CSS,  Typescript/JavaScript,  Karma/ Jasmine |
| Service End | Spring Boot |
| Database | Cloud Spanner |
| Source Control | GIT, GitHub, GitHub Actions for CI & CD. |
| Cloud | Google Cloud |
| Unit testing | Junit/Jasmine |

# Implementation Notes

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

|  |  |  |
| --- | --- | --- |
| Milestone -1 | 5 days | * Implement user-stories using Angular framework * Design application with minimum backend or mock backend as the focus for milestone-1 in on frontend skills * Implement forms, data binding, validations * Use appropriate unit test framework |
| Milestone -2 | 10 days | **Spring Boot Web API:**  Implement Reservatoin service and User service by following below guidelines,  Create Spring Boot Web API REST Microservices to perform SAVE Operation using POST method.   * Using Microservices architecture * Follow coding standards * Follow standard project structure. * Log all request details. * Log errors. * 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 * Database connections and web service URLs should be configurable. * Use browser / POST Man to invoke APIs * Run SonarQube for code quality. * Implement Junit for unit testing. |
| Milestone -3 | 5 - days | * Integrate service layer with UI component. * Setup CI & CD pipelines. * Dockerize the application. |

# 9.0 Evaluation rubrics

|  |  |
| --- | --- |
| Angular | * Associate must have used angular components, modules, data binding, data validation, CLI commands * Associate must have used forms and forms validation * Associate must have used directives * Associate must have developed reusable components * Associate must have followed coding standards |
| REACT | * Associate must have used component, databinding, data validation, CLI commands. * Associate must have used forms and forms validation * Associate must have defined react state * Associate must have followed coding standards |
| Microservices | * REST controller * Follow controller -> service -> Dao model * Entity and model classes * Appropriate logging statements * Exception handling |
| Docker | * Dockerize the application * Build docker containers * Push your docker images to repository with the docker push command |