**ON DEMAND CAR WASH**

Customer microservice

POC  
Low Level Design (LLD)



By

**Avneet**

Date: 18/04/2022

**DOCUMENT APPROVAL**

**Approvers of this document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Department** | **Role** | **Signature** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**DOCUMENT CHANGE HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document**  **Version #** | **Author** | **Date** | **Description** |
|  | Avneet | 14/04/2022 | On Demand Car Wash  (Customer microservice) |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

***Document Purpose*** ***4***

***Intended Audience*** ***4***

***Project Background, Objective(s) 4***

**Project Background** **4**

**Project Objective 4**

**Tools and Technologies** **4**

***Design Pattern*** ***5***

***Solution Diagram*** ***5***

***Solution Steps*** ***6***

***Classes/Function*** ***7***

***Validation*** **8**

**Data model/table 9**

***API Canvas*** ***9***

***HTTP Status Code*** ***10***

***Unit Testing 10***

***Request*** ***13***

***Response 14***

# **Document Purpose**

This document describes the solution architecture for Customer Microservices

# **Intended Audience**

This document is intended as a reference for the following roles and stakeholders who are interested in the On Demand Car Wash microservices technical architecture.

|  |  |
| --- | --- |
| Role | Nature of Engagement of Hotel Management Portal Technical Architecture |
| Product Owners | Key stakeholder to ensure that the architecture is aligned with business goals. |
| Business Analysts | Business analysts are one of the stakeholders who are informed with the key architectural decisions. |
| Enterprise Architects | To enforce Hotel management Platform Architecture is aligned to business goals and architecture, architectural guidelines. |
| Solution Architects | To ensure solution design and architecture is aligned to business requirements, architectural guidelines. |
| Developers | Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with Hotel management Microservices |

# **Project Background, Objective(s)**

Project Background: Customer microservice leads to perform Management of admin details where one can register themselves and perform various operations.

Project Objective: Customer Microservice will perform various operations like listing, creation, updation, and deletion of Customer Details. User can first register themselves and then they can perform all the operations.

# **Tools and Technologies**

* Technology: Java
* Framework: Springboot , Spring- security
* Database: MongoDb
* Server: Eueka

# **Design Pattern**

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | API | Using HTTP requests, we will use the respective action to trigger various operations |

Solution Diagram

[Microservices]                                 [Model classes]                                               [Databases]

API GATEWAY

Port no.:1111

Eureka

Server-8761

customers

Customer

Microservices

(8080)

Customer

ReviewnRatings

orders

ReviewAndRatings

Order

# **Solution Steps**

**Customer Registration**

* Firstly, when customer/user enter this microservice he/she will the register themselves
* Here I have created a class name CustomerController() which is the controller class for this microservices .
* In AdminController() class I have added multiples methods like savecustomer (),authCustomer (),, findaallCustomers(),getCustomerById(),updateCustomer () and deleteCustomer ().
* Here I am going to talk about saveCustomermethod.
* We can simply say that this is the registeration part for customer/user. This is the post method where client have to enter the cId, cName, cphoneno, cEmail,cPassword;
* After filling these details the client have to submit the details.

**Customer Listing**

* This is the get method which was present in the Customer Controller() class.
* In this method we can see the details of orders done by customer
* We just have to change to request method from post to get in postman and send the request.
* The server checks the details which are present in to our mongodb server and show the available details which are present.
* There are two different get request present in the Customer Controller, first method show all the details present in the DB and second method only show the particular detail like by status

**Customer Update**

* This method is use to update the update order which are present in the mongodb database.
* Firstly, client have to send the put request by using the customer id in the postman.
* Postman receive the request of client and give him rights to update the details for the particular Customer Id.
* When client update the details he/she had to re-send the request or submit the updated request.
* After send the request the server receive the request and update the previous details with new details into the mongoDb database

# **Classes/function**

|  |  |
| --- | --- |
| **Class** | **Descriptions** |
| Customer.java | This is the model class. |
| Order.java | This is the model class. |
| ReviewAndRatings.java | This is the model class. |
| CustomerRepository.java | This class deals with the data accessibility for guest and save data to mongoDb. |
| CustomerServiceImp.java | This is the service class for Customer microservice. |
| ApiRequestException.java | This class is use to handle all the api exception method. |
| CustomerServiceImpl.java | This is the service class for our guest microservice here we will write the implementation logic. |
| CustomerController.java | This is the main controller class where we will define that which service will class which request is send this will handle all the controller section for this Customer microservice |
| CustomerReviewController.java | This is the main controller class where we will define that which service will class which request is send this will handle all the controller section for this Customer microservice in which deal with customer reviews. |
| CustomerApplication.java | This is the main class where we run our Customer microservice. |

# **Validations**

FormValidation

* Customer Name---must not be empty max size should be 20
* Customer email---must contain @ symbol
* Password =should not empty

# **Data model/Table**

|  |  |
| --- | --- |
| **Data Type name** | **Type of data** |
| Private cName | String |
| Private cPhone | String |
| Private cEmail | String |
| Private cPassword | String |

# **API canvas**

|  |  |  |  |
| --- | --- | --- | --- |
| Micro Service | Path | Type | API Description |
| Customer service | /addcustomer | POST | Add the customer |
| Customer service | /allcustomers | GET | Get all customer list |
| Customer service | /allcustomers/{id} | GET | Get only that customer details whose id match |
| Customer  service | / update/{id} | PUT | Update the user with cid |
| Customer  service | /delete/{id} | DELETE | Delete the user with cid |
| Customer service | /getallorders/{id} | GET | Get only that order details whose id match |
| Customer  service | /addorder | POST | Add the order |
| Customer  service | /cancelorder/{id} | DELETE | Cancel the order using cid |

# **HTTP Status Code**

201 – Customer Registered

200 - Request succeeded

400 – Inputs are invalid

404 – Customer Not found

502 – Bad gateway

# **Unit Testing**

|  |  |
| --- | --- |
| Project Name | Customer Microservice |
| Created by | Avneet kaur chahal |
| Date of Creation | 18/04/2022 |
| Date of review |  |

## **For Registration of admin**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| TC\_o1 | Customer  registration | Enter the valid data to get registered | Customer needs to enter all the valid details | 1. Enter   Id:2  Customer\_name: John  email:avneet@gmail.com  phone:678900005  password:fghjkl  2) Enter Submit | <Valid Details> | Successful registration | Successful registration |
| TC\_o2 | Customer registration | Enter invalid data to get registered | Customer needs to enter the valid details with email | 1. Enter   Id:4  Admin\_name: John  email:avneetgmail.com  phone no:56789  2) Enter Submit | <invalid phone number> | Invalid  registeration | Email should have @  In email address  Phone number should be 10 digits |
| TC\_o3 | Admin registration | Enter all the required fields to get registered | Customer must enter all the required field as per specified in schema | 1) Enter  Id:2  Customer\_name: John  email:avneet@gmail.com  phone:678900005  password:fghjkl  2) Enter Submit | <All the Required fields are available> | Successful registration | Successful registration |
| TC\_o4 | Customer registration | Enter all the required fields to get registered | If customer misses one of the fields during registration which is marked as required in schema | 1) Enter  Id:8  admin\_name: John  phone:4567  password:fghjkl  2) Enter Submit | <email is missing> | You need to enter email | You need to enter email |

## **For customer listing**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULTS |
| TC\_o1 | Customer List | enter the valid admin Id of admin in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Customer Details | Customer Details |
| TC\_o2 | Customer List | enters the wrong admin Id which is not there in database to get admin details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Customer Details with this id is not present | Customer Details with this id is not present |

## For Customer Updating

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS(PASS/FAIL) |
| TC\_o1 | Customer Updating | | enter the valid id of admin in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid  Order  status | Customer Update  Successfully | CustomerUpdate  Successfully | pass |
| TC\_o2 | Customer Updating | enters the wrong id which is not there in database to get customer details | | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid order id | Customer Details with this  Customer Id you want to update is not present | Customer Details with this id you want to update is not present | pass |

# **Request**

**Register Customer**

/addcustomer {

Id:5

admin\_name: John

email:avneet@gmail.com

password:fghjkl

}

**Update Customer**

/ update/{id}{

“

cId:34”

}

# Response:

## **Register Customer**

**If valid details**

{

    "message": "Customer registered successfully"

}

status code: 201

**If invalid details**

{

“message”: “Inputs are not valid”

}, status code: 400

**If server encounters unexpected error**

{

  “message” :” Internal server error ”

}, status code: 500

## **List Customer**

**If valid ID**

{

   “id”: “45”,}

status code: 200

**If invalid ID**

{

“message”: “Customer not found”

}, status code:404

**If server encounters unexpected error**

{

  “message” :” Internal server error ”

}, status code: 500

Documented by Avneet kaur chahal