ON DEMAND CAR WASH SYSTEM

**LOW LEVEL DESIGN (LLD)**



**Date: 15-04-2022**

**Current Document Version: 1.0**

**DOCUMENT APPROVAL**

**Approvers of this document**

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

**DOCUMENT CHANGE HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version #** | **Author** | **Date** | **Description** |
| 1.0 | Pendyala Gowtham | 19-04-2022 | On demand Car Wash System Microservices |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[OVERVIEW 3](#_Toc101282641)

[DOCUMNT PURPOSE 3](#_Toc101282642)

[INTENDED AUDIENCE 3](#_Toc101282643)

[PROJECT BACKGROUND, OBJECTIVE 3](#_Toc101282644)

[Project Objective: 4](#_Toc101282645)

[Project Background: 4](#_Toc101282646)

[MICROSERVICES: 4](#_Toc101282647)

[TECHNOLOGY: 4](#_Toc101282648)

[DESIGN PATTERN: 4](#_Toc101282649)

[SOLUTION DIAGRAM 5](#_Toc101282650)

[SOLUTION STEPS 6](#_Toc101282651)

[REGISTRATION: 6](#_Toc101282652)

[UPDATION: 7](#_Toc101282653)

[DELETION: 8](#_Toc101282654)

[HTTP STATUS CODE: 8](#_Toc101282655)

[VALIDATIONS 9](#_Toc101282656)

[CLASSES/FUNCTIONS: 9](#_Toc101282657)

[CLASSES IN ADMIN MICROSERVICE 9](#_Toc101282658)

[CLASSES IN USER MICROSERVICE: 10](#_Toc101282659)

[CLASSES IN WASHER MICROSERVICE: 10](#_Toc101282660)

[CLASSES IN ORDER MICROSERVICE: 11](#_Toc101282661)

[CLASSES IN PAYMENT MICROSERVICE: 11](#_Toc101282662)

[CLASSES IN USERSENDMSG MICROSERVICE: 12](#_Toc101282663)

[CLASSES IN WASHERSENDMSG MICROSERVICE: 12](#_Toc101282664)

[CLASSES IN USERRECEIVEMSG MICROSERVICE: 12](#_Toc101282665)

[DATA MODELS/TABLES: 13](#_Toc101282666)

[API CANVAS 14](#_Toc101282667)

[UNIT TESTING 15](#_Toc101282668)

[FOR ADDING ADMINS 15](#_Toc101282669)

[FOR ADDING USERS 16](#_Toc101282670)

[FOR ADDING WASHER 17](#_Toc101282671)

[FOR ADMIN LISTING 17](#_Toc101282672)

[FOR WASHER LISTING 18](#_Toc101282673)

[REQUESTS 18](#_Toc101282674)

[ENV VARIABLES 22](#_Toc101282675)

# OVERVIEW

Every Car owner wishes to do car wash but doesn’t want to get his/her hands dirty doing it by washing their cars. It is tedious, time consuming and anybody would love not to do. Generally if you are planning to take your car to wash you need to plan it atleast an hour before and you need to sacrifice a half day of your work to complete the car wash. But it’s an awkward task. To solve this we came up with a solution and that is Green wash Company where our washers come into your place and wash your car at your specified time

The on demand car wash app is now become the users primary choice when planning for a car wash

# DOCUMENT PURPOSE

This document describes the solution architecture for on demand Car wash system microservice.

# INTENDED AUDIENCE

This document is intended as a reference for the following roles who are interested in on demand car wash system architecture

# PROJECT BACKGROUND, OBJECTIVE

## Project Objective:

The main objective of this On demand car wash system is when user enters car details and location he can get his car washed according to his convenience. It may be his home, office whatever it can be. Which can save his/her valuable time

## Project Background:

On demand car wash system makes user to login and perform various operations like add user, delete user, give ratings, schedule time, login and generate bills

# MICROSERVICES

Admin, User, Washer, Orders, Payment, Ratings, Plans, Userreievemessage, Usersendmessage, Washersendmessage, Washerrecievemessage.

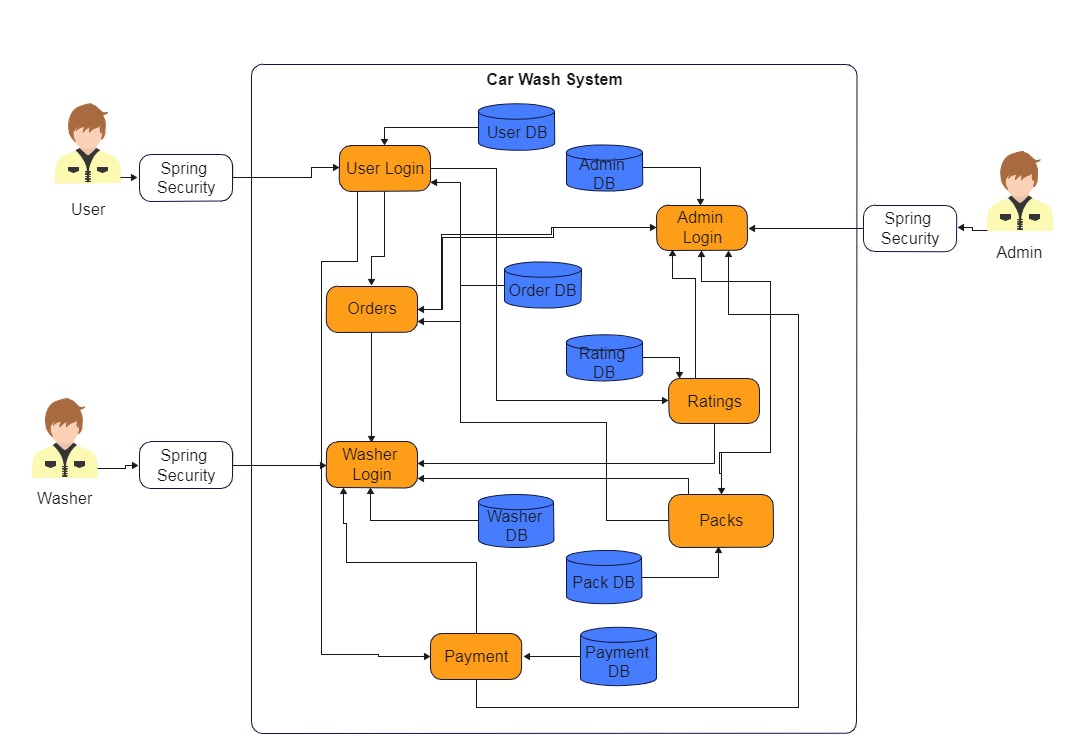
# TECHNOLOGY

Spring, Eureka, Swagger, RabbitMQ, MongoDB.

# DESIGN PATTERN

|  |  |  |
| --- | --- | --- |
| S/No | Name | Description |
| 1 | ADMIN MICROSERVICE | Using HTTP requests, we will perform various operations such as CRUD operations on plans and orders |
| 2 | USER MICROSERVICE | Using this microservice we will perform the basic CRUD operations like login, add user, plans and ratings |
| 3 | WASHER MICROSERVICE | By this microservice we will perform the operations like orders, add washer, generate bills, and ratings |
| 4 | ORDERS | By this microservice we will do the operations like taking the order and scheduling time |
| 5 | PAYMENT | We will perform CRUD operations of payment that is add payment and delete payment |
| 6 | USERRECIEVEMESSAGE | In this microservice we will use RabbitMQ to send the receive the message to the user |
| 7 | USERSENDMESSAGE | In this service we will use RabbitMQ to send the message from the user to washer |
| 8 | WASHERSENDMESSAGE | In this we will use the technology RabbitMQ to send the message from the washer to the user |
| 9 | WASHERRECIEVEMESSAGE | In this we will receive the message from the user to washer using RabbitMQ |

# SOLUTION DIAGRAM

****

# SOLUTION STEPS

## REGISTRATION:

1)Three end users: a) Washer

b) Customer

c) Admin

Need to be registered by entering the required details such as firstName, lastName, email, phone number, gender and click the submit button browser redirect the request to registration API

2) Call reaches the API gateway

3) API gateway does the routing and forwards the request to handler and this handle function will call the process

4) Process will call the validation to do the input validation

a) If validation fails then it will return error code and error description

with status code

b) If Validation successful then the handler will call the service which will

store data in database

5) It sends a response body with HTTP success response to handler

6) Handler returns JSON response

7) Success JSON response and HTTP status code with success Message

## UPDATION:

1) Three end users: a) Washer

b) Customer

c) Admin

Can update the details enters the id and details which end user wants to update the details. Browser directs the request to Update API

2) Call reaches the API gateway

3) API gateway does the routing and forwards the request to handler and this handle function will call the process

4) Process will call the validation to do the input validation

a) If validation fails then it will return error code and error description

with status code

b) If Validation successful then the handler will call the service which will

store data in database

5) It sends a response body with HTTP success response to handler

6) Handler returns JSON response

7) Success JSON response and HTTP status code with corresponding success message

## DELETION:

1) Three end users: a) Washer

b) Customer

c) Admin

Can delete the details enters the id and details which end user wants to update the details. Browser directs the request to delete the API

2) Call reaches the API gateway

3) API gateway does the routing and forwards the request to handler and this handle function will call the process

4) Process will call the validation to do the input validation

a) If validation fails then it will return error code and error description with

status code

b) If Validation successful then the handler will call the service which will

delete the data and update the data base

5) It sends a response body with HTTP success response to handler

6) Handler returns JSON response

7) Success JSON response and HTTP status code with corresponding success message

# HTTP STATUS CODE

400 : INVALID

404 : NOT FOUND

200 : SUCCEEDED

201 : REGISTERED

500: INTERNAL SERVER ERROR

# VALIDATIONS

****

# CLASSES/FUNCTIONS:

## CLASSES IN ADMIN MICROSERVICE

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | Main class having springbootapplication annotation |
| 2 | ADMINCONTROLLER.JAVA | This class contains all endpoints present in microservice |
| 3 | ADMINDETAILS.JAVA | We declare variables for admin details in this class |
| 4 | RATING.JAVA | We declare rating variables for rating details in this class |
| 5 | WASHPACKS.JAVA | We declare variables for pack details in this class |
| 6 | ADMINREPO.JAVA | This class is used to connect admin database to service |
| 7 | RATINGREPO.JAVA | This class is used to connect rating database to service |
| 8 | WASHPACKSREPO.JAVA | This class is used to connect packages database to service |
| 9 | ADMINTESTING.JAVA | This class is for unit testing |

## CLASSES IN USER MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | Main class having springbootapplication annotation |
| 2 | USERCONTROLLER.JAVA | This class having all end points |
| 3 | ORDERS.JAVA | This model class have all order details |
| 4 | PAYMENTS.JAVA | This model class having all the payment details |
| 5 | RATING.JAVA | We declare variables for rating details in this rating class |
| 6 | USER.JAVA | We declare variables for users in this class |
| 7 | WASHPACKS.JAVA | We declare variables for pack details in this class |
| 8 | USERREPO.JAVA | This class is used to connect user database to service |
| 9 | USERSERVICE.JAVA | This class binds user details data with variables |
| 10 | SECURITYCONFIG.JAVA | This class is used to configure jwt security to the service |
| 11 | JWTFILTER.JAVA | In this class we configure the filter used in security |
| 12 | USERDETAILS.JAVA | This class is used to extend user details service to set username and password |
| 13 | JWTUTIL.JAVA | This class is used for creating and validating jwt tokens |

## 

## CLASSES IN WASHER MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | Main class having springbootapplication annotation |
| 2 | WASHERCONTROLLER.JAVA | This class having all end points |
| 3 | ORDERS.JAVA | This model class have all order details |
| 4 | RATING.JAVA | We declare variables for rating details in this rating class |
| 5 | WASHER.JAVA | We declare variables for washer in this class |
| 6 | WASHPACKS.JAVA | We declare variables for pack details in this class |
| 7 | WASHERREPO.JAVA | This class is used to connect washer database to service |
| 8 | WAHSERSERVICE.JAVA | This class binds washer details data with variables |
| 9 | SECURITYCONFIG.JAVA | This class is used to configure jwt security to the service |
| 10 | JWTFILTER.JAVA | In this class we configure the filter used in security |
| 11 | USERDETAILS.JAVA | This class is used to extend washer details service to set username and password |
| 12 | JWTUTIL.JAVA | This class is used for creating and validating jwt tokens |

## CLASSES IN ORDER MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | Main class having springbootapplication annotation |
| 2 | ORDERCONTROLLER.JAVA | This class having all end points |
| 3 | ORDERDETAILS.JAVA | This model class have all order details |
| 4 | ORDERREPO.JAVA | This class is used to connect order database to service. |
| 5 | ORDERSERVICE.JAVA | This class binds data of orders to its variables. |

## 

## CLASSES IN PAYMENT MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | Main class having springbootapplication annotation |
| 2 | PAYMENTCONTROLLER.JAVA | This class having all end points |
| 3 | PAYMENTDETAILS.JAVA | This model class have all variables for order details |
| 4 | PAYMENTSREPO.JAVA | This class is used to connect payments database to service. |
| 5 | PAYMEBTSSERVICE.JAVA | This class binds data of payments to its variables. |

## CLASSES IN USERSENDMSG MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | This class contain overridden run method to configure what message to send. |
| 2 | USERCONFIG.JAVA | In this class we create all the required beans like queue, binding, connectionFactory etc. |

## CLASSES IN WASHERSENDMSG MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | This class contain overridden run method to configure what message to send. |
| 2 | USERCONFIG.JAVA | In this class we create all the required beans like queue, binding, connectionFactory etc. |

## CLASSES IN USERRECEIVEMSG MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | main class having springbootapplication annotation. |
| 2 | USERRECEIVEMSGCONFIG.JAVA | In this class we create all the required beans like queue, binding, connectionFactory etc. |
| 3 | JMSRECEIVER.JAVA | Using this class we print the received message in console. |

CLASSES IN WASHERRECEIVEMSG MICROSERVICE:

|  |  |  |
| --- | --- | --- |
| S/No | CLASS | DESCRIPTION |
| 1 | APPLICATION.JAVA | main class having springbootapplication annotation. |
| 2 | WASHERRECEIVEMSGCONFIG.JAVA | In this class we create all the required beans like queue, binding, connectionFactory etc. |
| 3 | JMSRECEIVER.JAVA | Using this class we print the received message in console. |

# DATA MODELS/TABLES

|  |  |
| --- | --- |
| Admin\_Id | INT |
| Admin\_Name | Varchar(25) |
| Admin\_Phone | Varchar(10) |
| Admin\_Gmail | Varchar(255) |
| Admin\_Address | Varchar(255) |
| User\_Id | INT |
| User\_Name | Varchar(25) |
| User\_Phone | Varchar(10) |
| User\_Gmail | Varchar(255) |
| User\_Address | Varchar(255) |
| Washer\_Id | INT |
| Washer\_Name | Varchar(25) |
| Washer\_Phone | Varchar(10) |
| Washer\_Gmail | Varchar(255) |
| Washer\_Address | Varchar(255) |
| Car\_Name | Varchar(255) |
| Car\_Model | Varchar(255) |
| WashPack\_Id | INT |
| Rating | INT |
| Review | Varchar(255) |

# API CANVAS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MICRO  SERVICE | PATH | VERB | DESCRIPTION | ROLE | AUTH |
| Admin | /addadmin | post | Admin registration | No | True |
| Admin | /alladmins | get | To get the list of admins | Admin | True |
| Admin | /alladmins/{id} | get | To get the list of admins based on Id | Admin | True |
| Admin | /addpack | post | Adding the pack into database | No | True |
| Admin | /allpacks | get | To get the list of packs | Admin | True |
| Admin | /delete/{id} | delete | To delete the pack from data base | Admin | True |
| Admin | /addrating | post | To add the rating to the data base | No | True |
| Admin | /allrating | get | To get the list of ratings from data base | Admin | True |
| Orders | /addorder | post | To add the order to the database | No | True |
| Orders | /allorders | get | To get the list of orders from data base | Admin | True |
| Orders | /delete/{id} | delete | To delete the order from database | Admin | True |
| Users | /adduser | post | To add the user to the data base | No | True |
| Users | /allusers | get | To get the list of users from data base | User | True |
| Users | /Update/{id} | put | To update the users details | User | True |
| Users | /allusers/{id} | get | To get all users by Id | User | True |
| Users | /delete/{id} | delete | To delete the users by Id | User | True |
| Washer | /addwasher | post | To add the washer to the data base | No | True |
| Washer | /allwasher | get | To get the all washers from the data base | Washer | True |
| Washer | /allwasher/{id} | get | To get the washer based on the id | Washer | True |
| Washer | /delete | delete | To delete the washer from the data base | Washer | True |

# UNIT TESTING

|  |  |
| --- | --- |
| Project Name | On demand Car Wash System |
| Created by | Group 1 |
| Date of creation | 19/04/2022 |
| Date of review |  |

## FOR ADDING ADMINS:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | SCENERIO | TEST CASE | PRE  CONDITION | STEPS | DATA | EXPECTED | ACTUAL |
| 1 | Adding Admin | Enter the valid details to add the Admin | Admin needs to enter all the valid details | 1. Enter id:1   Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi   1. Enter Submit | <Valid Details> | Admin added successfully | Admin added successfully |
| 2 | Adding  Admin | Enter the invalid details to add the Admin | Admin needs to enter the valid details | 1. Enter id:1   Name: john  Phone:0123459  Gmail:abc@gmail  Address: Delhi   1. Enter Submit | <Invalid  Details> | Phone number should be of  10 digits | Phone number should be of  10 digits |
| 3 | Adding Admin | Enter all the required  Fields to get  registered | Admin must enter all the required fields as per specified | 1. Enter id:1   Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi   1. Enter Submit | <All fields are available  > | Admin added  successfully | Admin added  successfully |
| 4 | Adding Admin | Enter all the required  fields to get registered | If the admin miss one of the fields which is marked as required | 1. Enter id:1   Name: john  Phone:0123456789  Address: Delhi   1. Enter Submit | <email is missing> | You need to enter email | You need to enter email |

## FOR ADDING USERS:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | SCENERIO | TEST CASE | PRE  CONDITION | STEPS | DATA | EXPECTED | ACTUAL |
| 1 | Adding User | Enter the valid details to add the User | User needs to enter all the valid details | 1)Enter id:1  Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi  2)Enter Submit | <Valid Details> | User added successfully | User added successfully |
| 2 | Adding  User | Enter the invalid details to add the User | User needs to enter the valid details | 1)Enter id:1  Name: john  Phone:0123459  Gmail:abc@gmail  Address: Delhi  2)Enter Submit | <Invalid  Details> | Phone number should be of  10 digits | Phone number should be of  10 digits |
| 3 | Adding User | Enter all the required  Fields to get  registered | Admin must enter all the required fields as per specified | 1. Enter id:1   Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi  2)Enter Submit | <All fields are available  > | User added  successfully | User added  successfully |
| 4 | Adding User | Enter all the required  fields to get registered | If the User miss one of the fields which is marked as required | 1)Enter id:1  Name: john  Phone:0123456789  Address: Delhi  2)Enter Submit | <email is missing> | You need to enter email | You need to enter email |

## FOR ADDING WASHER:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | SCENERIO | TEST CASE | PRE  CONDITION | STEPS | DATA | EXPECTED | ACTUAL |
| 1 | Adding Washer | Enter the valid details to add the Washer | Washer needs to enter all the valid details | 1)Enter id:1  Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi  2)Enter Submit | <Valid Details> | Washer added successfully | Washer added successfully |
| 2 | Adding  Washer | Enter the invalid details to add the Washer | Washer needs to enter the valid details | 1)Enter id:1  Name: john  Phone:0123459  Gmail:abc@gmail  Address: Delhi  2) Enter Submit | <Invalid  Details> | Phone number should be of  10 digits | Phone number should be of  10 digits |
| 3 | Adding Washer | Enter all the required  Fields to get  registered | Washer must enter all the required fields as per specified | 1)Enter id:1  Name: john  Phone:0123456789  Gmail:abc@gmail  Address: Delhi  2)Enter Submit | <All fields are available  > | Washer added  successfully | Washer added  successfully |
| 4 | Adding Washer | Enter all the required  fields to get registered | If the Washer miss one of the fields which is marked as required | 1)Enter id:1  Name: john  Phone:0123456789  Address: Delhi  2)Enter Submit | <email is missing> | You need to enter email | You need to enter email |

## FOR ADMIN LISTING:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | SCENERIO | TEST CASE | PRE  CONDITION | STEPS | DATA | EXPECTED | ACTUAL |
| 1 | Admin List | Enter the valid id of admin to get the details | That id needs to be present in data base | 1. Enter valid id 2. Enter submit | Valid id | Admin Details | Admin Details |
| 2 | Admin List | Enter the wrong id which is not there in database to get admin details | That id needs to be present in data base | 1. Enter invalid id | Invalid id | Admin details with this id is not present | Admin details with this id is not present |

## FOR WASHER LISTING:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | SCENERIO | TEST CASE | PRE  CONDITION | STEPS | DATA | EXPECTED | ACTUAL |
| 1 | Washer List | Enter the valid id of washer to get the details | That id needs to be present in data base | 1)Enter valid id  2)Enter submit | Valid id | Washer  Details | Admin Details |
| 2 | Washer List | Enter the wrong id which is not there in database to get washer details | That id needs to be present in data base | 1)Enter invalid id  2) Enter submit | Invalid id | Washer  details with this id is not present | Washer details with this id is not present |

# REQUESTS

**/user/adduser ->**

{

"id": 1,

"location": "delhi",

"name": "sai",

"password": "sai",

"gmail": "sai@gmail.com",

"phoneNumber": "123456789"

}

**/user/update/{id} ->**

{

"id": 1,

"location": "Jangareddygudem",

"name": "sai",

"password": "sai",

"gmail": "sai@gmail.com",

"phoneNumber": "123456789"

}

**/user/addorder ->**

{

"carModel": "string",

"carName": "string",

"date": 0,

"orderId": 0,

"phoneNo": 0,

"washerName": "string",

"washpackId": 0

}

**/user/addrating ->**

{

"comment": "string",

"rating": 0,

"washerName": "string"

}

**/admin/addadmin ->**

{

"id": 0,

"name": "string"

}

**/admin/addpacks ->**

{

"cost": 0,

"description": "string",

"id": 0,

"name": "string"

}

**/washer/addwasher ->**

{

"id": 1,

"location": "delhi",

"name": "sai",

"password": "sai",

"gmail": "sai@gmail.com",

"phoneNumber": "123456789"

}

**/washer/update/{id} ->**

{

"id": 1,

"location": "Jangareddygudem",

"name": "sai",

"password": "sai",

"gmail": "sai@gmail.com",

"phoneNumber": "123456789"

}

**RESPONSE ->**

**Get requests:**

If valid request

[

{

"id": 1,

"name": "sai",

"location": "jangareddygudem",

"password": "sai"

},

{

"id": 2,

"name": "jack",

"location": "jrg",

"password": "lol"

}

]

Status code:200

If invalid request ID

{

“message”: “Customer not found”

}, status code:404

Update requests

If valid ID

{

“Customer updated successfully with id: ”

}

status code: 200

If invalid ID

{

“message”: “CAN NOT UPDATE AS USER NOT FOUND WITH THIS ID :”

}, status code:404

If server encounters unexpected error

{

“message” :” Internal server error”

}, status code: 500

Delete requests

If valid ID

{

“Customer Deleted successfully”

}

status code: 200

If invalid ID

{

“message”: “CAN NOT DELETE AS USER NOT FOUND WITH THIS ID :”

}, status code:404

If server encounters unexpected error

{

“message” :” Internal server error ”

}, status code: 500

ENV VARIABLES

spring.application.name=USER-SERVICE

server.port=8080

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

spring.data.mongodb.uri=mongodb+srv://sai:sai@cluster1.66bfh.mongodb.net/test

spring.data.mongodb.database=user

spring.application.name=WASHER-SERVICE

server.port=8081

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

spring.data.mongodb.uri=mongodb+srv://sai:sai@cluster1.66bfh.mongodb.net/test

spring.data.mongodb.database=washer

spring.application.name=ADMIN-SERVICE

server.port=8082

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

spring.data.mongodb.uri=mongodb+srv://sai:sai@cluster1.66bfh.mongodb.net/test

spring.data.mongodb.database=admin

spring.application.name=ORSER-SERVICE

server.port=8083

spring.mvc.pathmatch.matching-strategy=ant-path-matcher

spring.data.mongodb.uri=mongodb+srv://sai:sai@cluster1.66bfh.mongodb.net/test

spring.data.mongodb.database=order