**Pharmacy Management system**

**POC**  
**Low Level Design (LLD)**



Date:

Current Document Version:

DOCUMENT APPROVAL

**Approvers of this document**

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

**Document Change History**

|  |  |  |
| --- | --- | --- |
| **Document Version #** | **Author** | **Description** |
| 1.0 | A. Neeraj Krishna | Pharmacy Management System LLD |
| 2.0 | A. Neeraj Krishna | Angular |
|  |  |  |
|  |  |  |

Table of Contents

[1.0 Document Purpose 4](#_Toc94636300)

[2.0 Intended Audience 4](#_Toc94636301)

[3.0 Project Background, Objective(s) 4](#_Toc94636302)

[3.1 Project Background 4](#_Toc94636303)

[3.2 Project Objective 4](#_Toc94636304)

[4.0 Design Pattern 5](#_Toc94636305)

[5.0 Solution Diagram 5](#_Toc94636306)

**5.1 Solution Diagram**

**5.2 ER Diagram**

**5.3 Use Case Diagram**

**5.4 Data Flow**

[6.0 Solution Steps 5](#_Toc94636307)

[7.0 Classes/function name 6](#_Toc94636308)

[8.0 Validations 6](#_Toc94636309)

[9.0 Data model/Tables 6](#_Toc94636310)

[10.0 API Canvas 6](#_Toc94636311)

**11.0 *User Requirements***

[12.0 Developer Requirments 7](#_Toc94636313)

[13.0 AWS Role 7](#_Toc94636314)

[14.0 HTTP Status Code 7](#_Toc94636314)

[15.0 Unit Testing 7](#_Toc94636314)

[16.0 Request 7](#_Toc94636314)

[17.0 Response 7](#_Toc94636314)

# Document Purpose

This document describes the solution architecture for Customer management microservice

# Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the Customer Management Microservice technical architecture.

|  |  |
| --- | --- |
| Role | Nature of Engagement in WB Classics Portal Technical Architecture |
| Product Owners/SME | 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 Customer 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 Customer management Microservice |

# Project Background, Objective(s)

## Project Background

Pharmacy management system leads to perform Management of Pharmacy where admin or doctor can login whereas admin some set of operations and doctor can do some set of operations.

## Project Objective

Pharmacy management system​ will perform various operations like listing, creation, updating and deletion of Drugs and suppliers, view all the orders the are placed by the doctors, payment process of verified orders and can get the sales report by printing and downloading options. The doctor can check the availability of drugs in pharmacy, place the order according to the requirements and buy the drugs that needed.

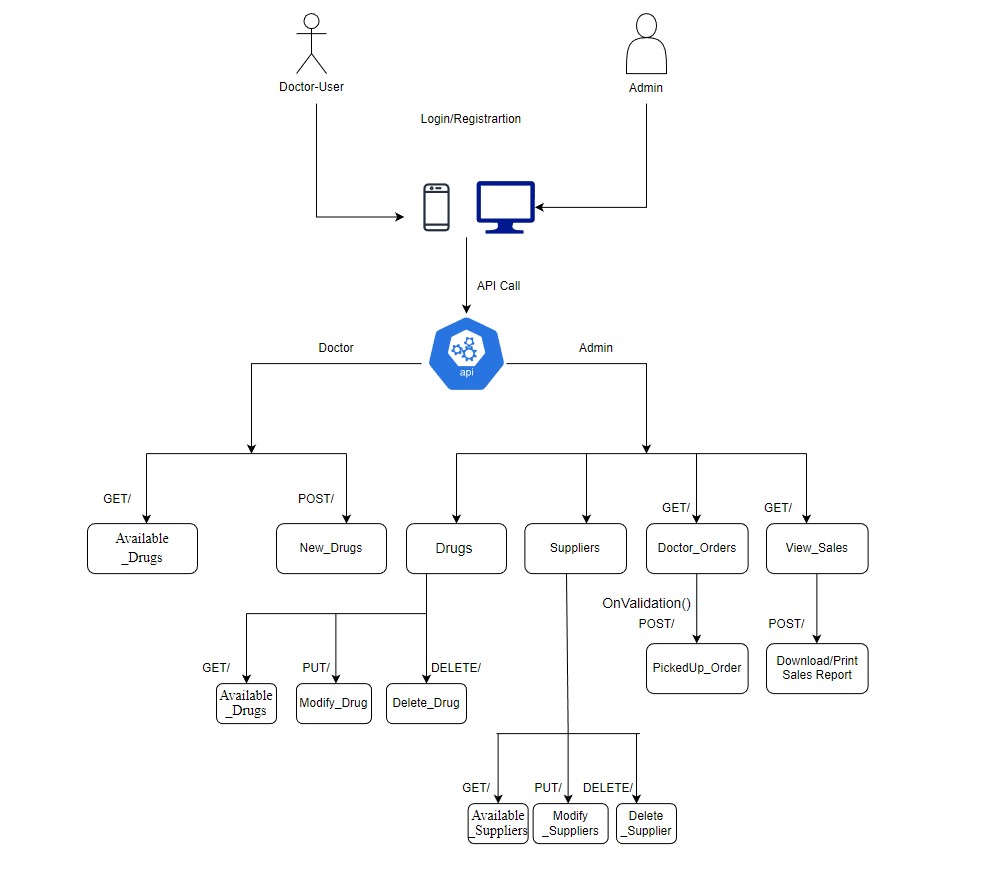
Admin/ Doctor can first register themselves and then they can perform all the operations.

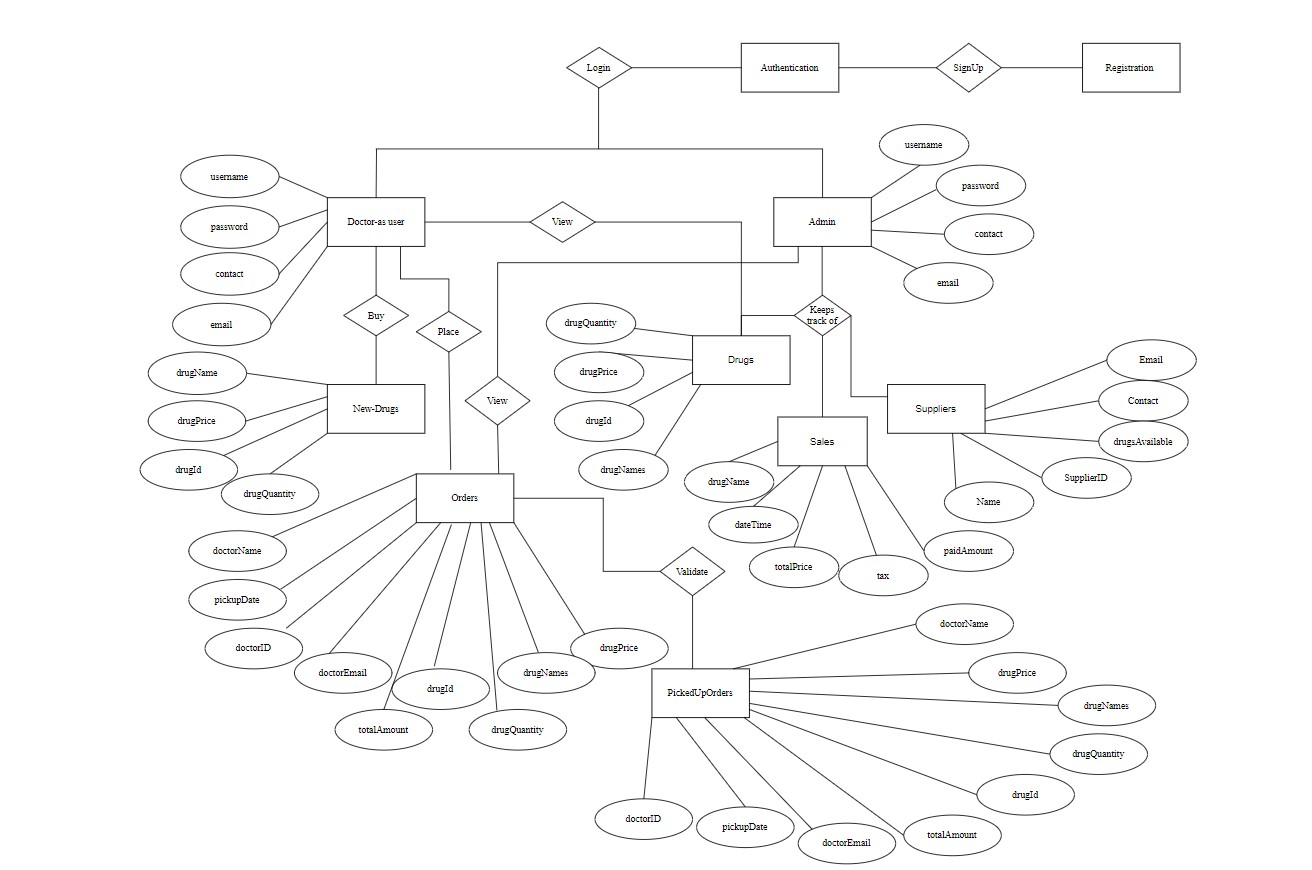
# Design Pattern

|  |  |  |
| --- | --- | --- |
| S.No | Name | Description |
| 1 | Angular | Using Angular we can create a single-page application with the help of Typescript and HTML |
| 2 | API | Using HTTP requests, we will use the respective action to trigger various operations |
| 3 | SQL | Used as Database for storing all the details like admin details drug details and doctor-order details |

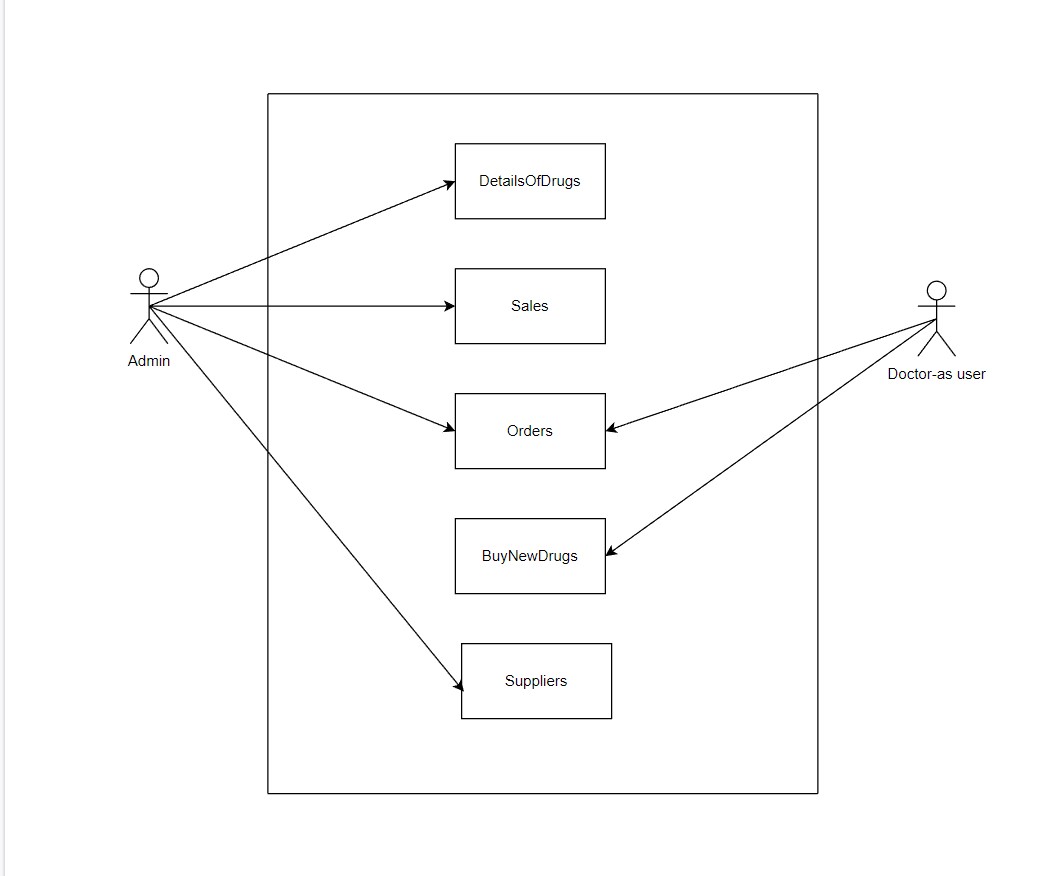
# Diagrams

5.1 Solution Diagram



5.2 ER Model: - 

5.3 Use Case Diagram:-



# 6.0 Solution Steps

**Admin Login**

1. The admin will select login and enter the required credentials .
2. There will be only one admin so, the end-user can’t signup as a admin.
3. The input validation will be done
4. If validation fails, then it will return the error code and error description. The JSON response and status HTTP will be returned.
5. If validation is successful, then the admin will get logged in and able to do the operations like, view, update, delete, create. The JSON response and status HTTP will be returned.

**Doctor Registration-as user**

1. The doctor can sign -in with the credentials that he had or he can register by giving the required constrance like email-Id, password, contact, name.
2. Doctor can view all the drugs that are available in the pharmacy.
3. Doctor will order the drugs if needed and then the order will be approved by the admin.

**Drugs Listing**

1. The Admin/Doctor will login using there credentials.
2. The Admin will be able to view all the drugs and admin can also make changes to the already existing drugs where as doctor will only be able view all the drugs that are present and can request a order of drugs if there is no stock for a drug.
3. The Order placed by the doctor will be verified by the admin and then admin will change the field that is valid or not.
4. Based on the verification of admin the order of the drug will be placed and then the JSON response and the status HTTP will be returned.

**Drugs Updation**

1. Admin wants to update the details enters the id and the details which admin wants to update the details. browser directs the request to customer update API
2. Call reaches the API gateway.
3. It sends response body with HTTP Success response code to updateCustomerHandler.
4. updateCustomerHandler returns JSON Response
5. Success JSON response and status HTTP code 200 with corresponding success message.

**Drugs Deletion**

1. Admin eneters the id in parameter for which customer wants to delete the details. browser directs the request to customer deletion API
2. Call reaches the API gateway.
3. doProcess () will call the deleteDrugsService.deleteDrugs() which calls the removeDrugsRepository.removeDrugs() to delete the data from database.
4. removeDrugsHandler returns JSON Response
5. Success JSON response and HTTP status code 200 with corresponding success message.

# 7.0 Classes/function

|  |  |  |
| --- | --- | --- |
| **#** | **Class** | **Description** |
| 1 | Admin.js | Model holds the Admin schema details |
| 2 | registerCustomerHandler.js | The handler to handle the registration of customers which calls the registerCustomerService class |
| 3 | Doctor-user.ts | Model holds the Doctor schema details |
| 4 | IDoctorRepository.ts | This class deals with the data accessibility for doctor registration |
| 5 | listHandler.js | The handler to handle the listing of customers. which calls the listCustomerService class |
| 7 | listDrugs.ts | This class deals with data accessibility for customer list |
| 8 | removeDrugsHandler.js | The handler to handle the deletion of customers. which calls the removeCustomerService class |
| 10 | IDrugRepository.ts | This class deals with data accessibility for Drug deletion |
| 11 | updateDrugHandler.js | The handler to handle the updation of Drug details in database which calls the updateDrugService class |
| 13 | IDrugRepository.ts | This class deals with data accessibility for customer deletion |
| 14 | OrderValidator.ts | It deals with the validation of the inputs provided by the Doctor-Order |

# Validations



# Data model/Table

|  |  |  |
| --- | --- | --- |
|  | DoctorOrders |  |
|  | doctorName | VARCHAR(25) |
|  | doctorContact | VARCHAR(25) |
| FK | doctorID | VARCHAR(25) |
|  | doctorEmail | VARCHAR(255) |
|  | drugsId | VARCHAR(25) |
|  | drugName | VARCHAR(255) |
|  | drugPrice | INTEGER |
|  | drugQuantity | VARCHAR (255) |
|  | realQuantity | INTEGER |
|  | totalAmount | VARCHAR(255) |
|  | pickupDate | DATE |

|  |  |  |
| --- | --- | --- |
|  | DoctorUser |  |
|  | doctorName | VARCHAR(25) |
|  | doctorContact | VARCHAR(25) |
| PK | doctorID | VARCHAR(25) |
|  | doctorEmail | VARCHAR(255) |
|  | password | VARCHAR(25) |
|  | dateTime | DATE TIME |

|  |  |  |
| --- | --- | --- |
|  | Inventory |  |
| FK | email | VARCHAR(25) |
|  | name | VARCHAR(25) |
|  | quantity | VARCHAR(25) |
| PK | batchId | VARCHAR(255) |
|  | expireDate | VARCHAR(25) |
|  | price | INTEGER |
|  | imagePath | VARCHAR(25) |

|  |  |  |
| --- | --- | --- |
|  | PickedUpOrders |  |
|  | doctorName | VARCHAR(25) |
|  | doctorContact | VARCHAR(25) |
| Fk | doctorID | VARCHAR(25) |
|  | doctorEmail | VARCHAR(255) |
|  | drugName | VARCHAR(255) |
|  | drugPrice | VARCHAR(10) |
|  | drugQuantity | VARCHAR (255) |
|  | realQuantity | INTEGER |
|  | totalAmount | VARCHAR(255) |
|  | pickupDate | DATE |
|  | dateTime | DATE TIME |

|  |  |  |
| --- | --- | --- |
|  | Sales |  |
|  | drugName | VARCHAR(255) |
|  | dateTime | DATA TIME |
|  | drugPrice | VARCHAR(10) |
|  | totalPrice | INTEGER |
|  | tax | VARCHAR(25) |
|  | totalAmount | VARCHAR(255) |
|  | balance | VARCHAR(25) |

|  |  |  |
| --- | --- | --- |
|  | Supplier |  |
| PK | supplierID | VARCHAR(25) |
|  | name | VARCHAR(25) |
|  | email | VARCHAR(25) |
|  | contact | VARCHAR(255) |
|  | drugsAvailable | VARCHAR(25) |
|  | Sales |  |
|  | drugName | VARCHAR(255) |
|  | dateTime | DATA TIME |
|  | | |
|  | Admin |  |
|  | name | INTEGER |
|  | contact | VARCHAR(25) |
| PK | email | VARCHAR(25) |
|  | password | VARCHAR(25) |
|  | role | VARCHAR(25) |
|  | dateTime | DATA TIME |

|  |  |  |
| --- | --- | --- |
|  | Verified Orders |  |
|  | doctorName | VARCHAR(25) |
|  | doctorContact | VARCHAR(25) |
| FK | doctorID | VARCHAR(25) |
|  | doctorEmail | VARCHAR(255) |
|  | drugName | VARCHAR(255) |
|  | drugPrice | VARCHAR(10) |
|  | drugQuantity | VARCHAR (255) |
|  | realQuantity | INTEGER |
|  | totalAmount | VARCHAR(255) |
|  | pickupDate | DATE |
|  | real Quantity | VARCHAR(25) |

# 10.0 API Canvas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Buy-New-Drugs | /Order | POST | To Order New Drugs | Doctor | True |
| Drug-Details | /AvailableDrugs | GET | To get Drug list | Admin | True |
| Update-DrugDetails | /DrugDetails/id | PUT | To update Drug details | Admin | True |
| Suppliers-Details | /Suppliers/id | GET | To get details of a particular Suppliers | Admin | True |
| Sales-Details | /Sales | GET | To get the Sales of the Pharmacy | Admin | True |
| Update-Supplier | /Supplier/Id | PUT | To update the existing supplier | Admin | True |

# 11.0 User Requirements

# 11.1 Hardware

# • Processor: Minimum 1.8 GHz. Recommended 2GHz or more.

# • Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

# • Hard Drive: Minimum 100 GB; Recommended 500GB or more.

# • Memory (RAM): Minimum 4 GB; Recommended 8 GB or above.

# • OS: Windows.

# 11.2 Software

# • Any Latest Browsers.

12.**0 Developer Requirements**

12.1 Hardware

• Processor: Minimum 1.8 GHz. Recommended 2GHz or more.

• Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

• Hard Drive: Minimum 100 GB; Recommended 500GB or more.

• Memory (RAM): Minimum 4 GB; Recommended 8 GB or above.

• OS: Windows.

12.2 Software

• Visual studio 2022.

•Sql server management studio (ssms).

• Node, Angular.

# 13.0 AWS Role

LAMBDA – RDS – Cloud Watch

LAMBDA Invocation

AWS API Gateway

# 14.0 HTTP Status Code

201 – Customer Registered

200 - Request succeeded

400 – Inputs are invalid

404 – Customer Not found

502 – Bad gateway

# 16.0 Unit Testing

|  |  |
| --- | --- |
| Project Name | Customer Management Microservice |
| Created by | Utsav Kumar |
| Date of Creation | 05/02/22 |
| Date of review | 07/02/22 |

**For Registration of customers**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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  customer\_first\_name: John  customer\_last\_name:doe  emai:customer@gmail.com  Phone:0123456789  age:22  address:delhi  gender:male  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 wrong phone number type | 1) Enter  customer\_first\_name: John  customer\_last\_name:do  emai:customer@gmail.com  Phone:01234567  age:22  address:delhi  gender:male  2) Enter Submit | <invalid phone number> | Phone number should be of 10 digits | Phone number should be 10 digits |
| TC\_o3 | Customer registration | Enter all the required fields to get registered | Customer must enter all the required field as per specified in schema | 1) Enter  customer\_first\_name: John  customer\_last\_name:doe  emai:customer@gmail.com  Phone:0123456789  age:22  address:delhi  gender:male  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  customer\_first\_name: John  customer\_last\_name:doe  Phone:0123456789  age:22  address:delhi  gender:male  2) Enter Submit | <email is missing> | You need to enter email | You need to enter email |

**For customers 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 Customer Id of customer 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 Customer 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 Id | Customer Details with this id is not present | Customer Details with this id is not present |

**For customers Deletion**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT |
| TC\_o1 | Customer Deletion | enter the valid id of customer in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Customer Deleted successfully | Customer Deleted successfully |
| TC\_o2 | Customer deletion | 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 Id | Customer Details with this id you want to delete is not present | Customer Details with this id you want to delete is not present |

**For customers Updation**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS(PASS/FAIL) |
| TC\_o1 | Customer Updation | enter the valid id of customer in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Customer Update  Successfully | Customer Update  Successfully | pass |
| TC\_o2 | Customer Updation | 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 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 |

# 16.0 Request

**Register Customer**

/customers

{

“customer\_first\_name”: “John”,

“customer\_last\_name” :”Doe”,

“email”:”customer@gmail.com”

“age” : 22,

“customer\_phone\_number” :” 0123456789”,

“gender”: “Male”,

“customer\_address”:”Jamshedpur”,

}

**Update Customer**

/customers/

{

“customer\_first\_name”: “Michael”,

“customer\_last\_name” :”Clarke”,

“email”:”customer@gmail.com”

“age” : 22,

“customer\_phone\_number” : 0123456789,

“gender”: “Male”,

“customer\_address”:”Jamshedpur”,

}

**17.0 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 customers**

**If valid ID**

{

“customer\_first\_name”: “Michael”,

“customer\_last\_name” :”Clarke”,

“email”:”customer@gmail.com”

“age”: 22,

“customer\_phone\_number” : 0123456789,

“gender”: “Male”,

“customer\_address”:”Jamshedpur”,

}

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

**Update customer**

**If valid ID**

{

“Customer updated successfully”

}

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

**Delete customer**

**If valid ID**

{

“Customer Deleted successfully”

}

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