**Hotel Management System**

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

****

Date:15/07/2022

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 | Debjit Chakraborty | 15/07/2022 | Hotel Management System LLD |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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)

[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](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636311)

[11.0 Env variables 6](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636312)

[12.0 Integrations 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636313)

[13.0 AWS Role 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636314)

[14.0 HTTP Status Code 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636314)

[15.0 Unit Testing 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636314)

[16.0 Request 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636314)

[17.0 Response 7](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636314)

# Document Purpose

This document describes the solution architecture for Hotel Management System.

# Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the Hotel Management System 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 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 Microservice |

# Project Background, Objective(s)

## ***Project Background***

Hotel Management System is to simplify the day to day processes of the hotel. The system will be able to handle many services to take care of all customers in a quick manner.

## ***Project Objective***

Hotel Management System will perform various operations like Make Reservations, search rooms, add payments, issue bills, manage guests, manage room details, manage staffs, manage inventories, manage users, E-mail notifications .

* 1. ***Hardware and software used***

**Hardware Requirements**

**Processor :** Core i3 and above

**Primary Memory :** 8GB

**Hard Disk :** 60GB free space

**Software Requirements**

**Operating System :** Windows XP or above

**Framework : Angular**

**Frontend :** HTML5, CSS3, JavaScript, Bootstrap

**Middleware :** ASP .net core Web API

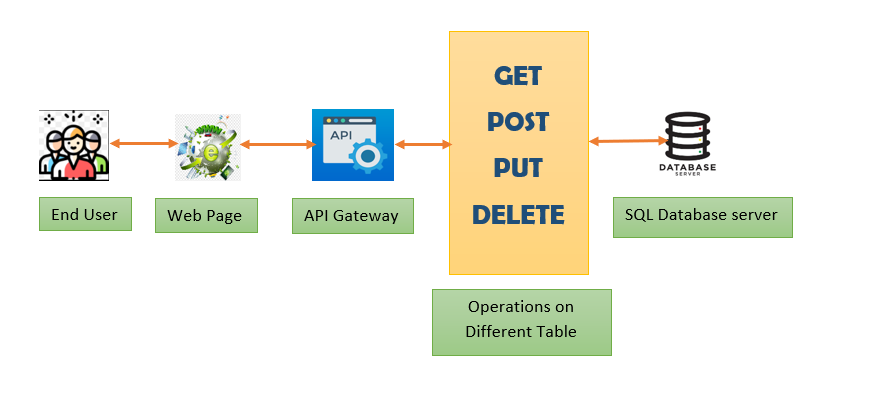
**Backend :** SQL Server

**Tools used :** Visual Studio, VS code, SSMS

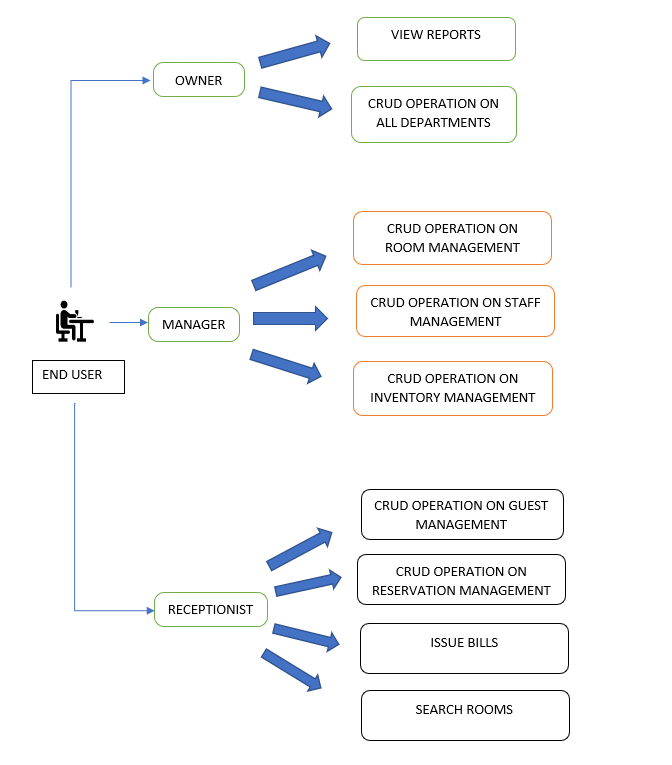
# Design Pattern

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | API | Using HTTP requests, we will use the respective action to trigger various operations |
| 2 | Angular | To create and design the front end. |
| 3 | Database | To store and retrieve the information. |

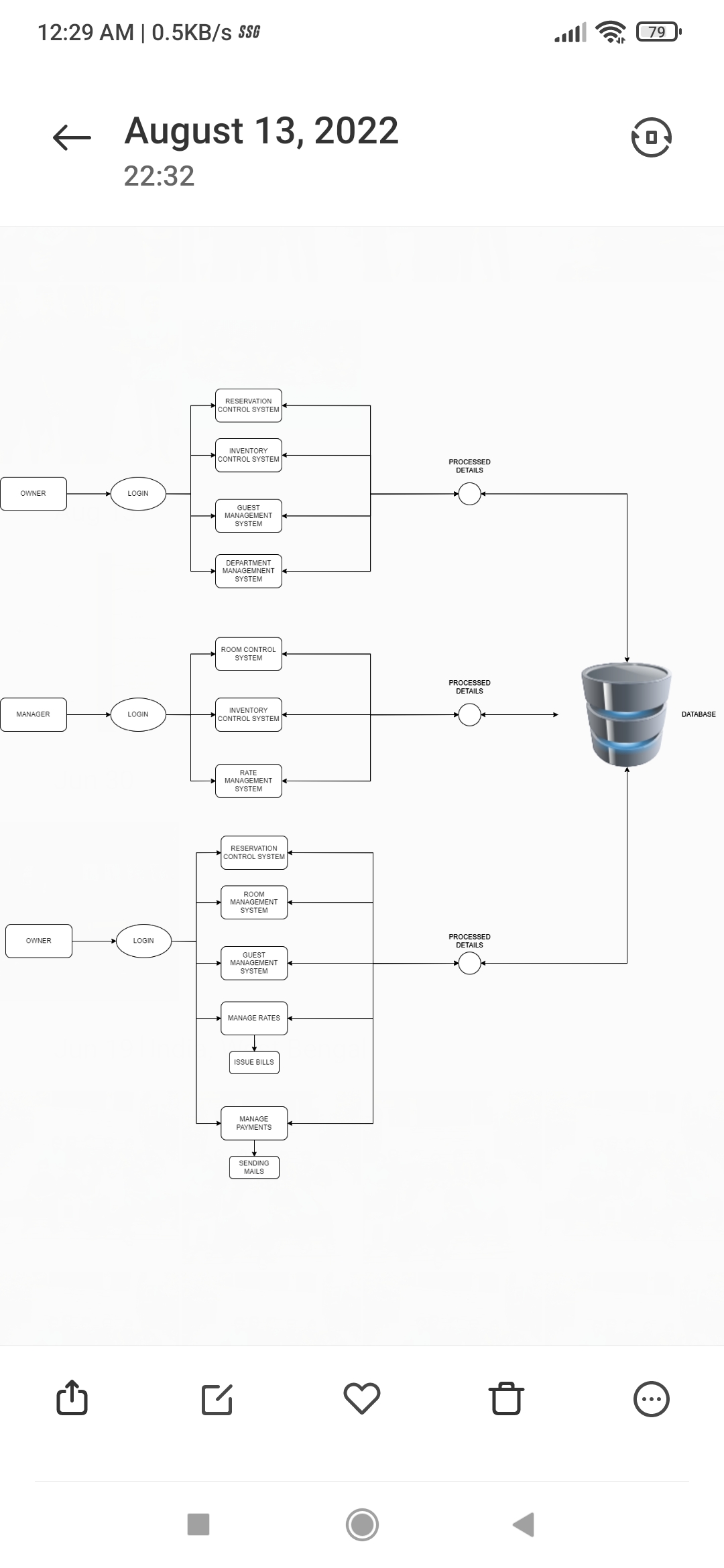
# Solution Diagram



# Use Case Diagram



**Flow** **Diagram**



# 6.0 Solution Steps

**Admin**

1. Admin will enter his login Id and password
2. As the admin clicks on the login button the browser directs the request to the admin login API.
3. Call reaches the API gateway.
4. API gateway does the routing to admin\_validate() function
   1. If the function returns true, the browser displays the hotel management dashboard.
   2. If the function returns false, the browser displays the error message

According to admin level permissions he/she can access the internal functionalities.

**Search Rooms**

1. The receptionist must enter details like Period, Check-in, Check-out, Guests and click search button.
2. When a match is found, a message along with the available rooms is displayed to the receptionist.

Based on the search details provided, the receptionist can make reservation for the customers

**Make Reservation**

1. Receptionist will enter the required customer details like Code, Number of children, Number of adults, check-in date, check out date, status, Number of nights and click submit button after which the browser directs the request to make reservation API
2. Call reaches the API gateway
3. API gateway does the routing and forwards to reservatioin\_validation() function.
   1. If the function returns true, the browser displays the dashboard.
   2. If the function returns false, the browser displays the error message
4. When the database is updated, the receptionist will be notified that the reservation is successful.

**Guest Details**

1. Receptionist will enter the required guest details like Member code, Phone number, Company, Name, E-mail, Gender, Address and click submit button after which the browser directs the request to add guest API
2. Calls the API gateway
3. Routing is processed by API gateway and forwards the request to the API controller which handles all the requests of adding guests.
4. API gateway does the routing to guestdetails\_validate() function
   1. If the function returns true, the browser displays the Guest dashboard.
   2. If the function returns false, the browser displays the error message
5. The receptionist can also modify the guest details if required and remove the guest details after check out of the guest.
6. When the database is updated, the receptionist will be notified that the guest is added/updated/deleted successfully.

**Inventory**

1. Manager will enter the required inventory details such as inventory name, quantity, price and click submit button after which the browser directs the request to add inventory
2. Calls the API gateway
3. Routing is processed by API gateway and forwards the request to the API controller which handles all the requests of adding inventory
4. Manager clicks on the Add button to add a new inventory details in the list as the request will pass to the addInventory() function.
5. Manager clicks on the Update button to update a inventory details in the list as the request will pass to the updateInventory() function.
6. Manager clicks on the Delete button to delete a inventory details from the list as the request will pass to the deleteInventory() function.
7. When the database is updated, the manager will be notified that the inventory is added/updated/deleted successfully.
8. Manager can also view a guest details by InventoryDetail button which will pass the request to getInventoryDetail() function.

**Staff Details**

1. Manager will enter the required guest details like Staff code, Employee name, Address, Salary, Age, Occupation, Email and click submit button after which the browser directs the request to add staff.
2. Calls the API gateway
3. Routing is processed by API gateway and forwards the request to the API controller which handles all the requests of adding a staff.
4. API gateway does the routing to staffdetails\_validate() function
   1. If the function returns true, the browser displays the Staff dashboard.
   2. If the function returns false, the browser displays the error message
5. The manager can also modify and remove the staff details if required.
6. When the database is updated, the manager will be notified that the guest is added/updated/deleted successfully.

**Issue Bill**

1. In order to issue bill for the customer, the receptionist must enter the required bill details such as Billing no, Quantity, Price, Taxes, Date, Services, Unit and click generate bill button.
2. All the fields entered by the receptionist will be validated.
3. In case of successful validation, the receptionist can print the bill by clicking print button.
4. If validation fails then appropriate message will be displayed and browser redirects to issue bill page.

# 7.0 Classes/function

|  |  |  |
| --- | --- | --- |
| **#** | **Class** | **Description** |
| 1 | Staff.cs | Model holds the Staff schema details |
| 2 | Inventory.cs | Model holds the Inventory schema details |
| 3 | Guest.cs | Model holds the Guest schema details |
| 4 | Room.cs | Model holds the Room schema details |
| 5 | Reservation.cs | Model holds the Reservation schema details |
| 6 | Rate.cs | Model holds the Rate schema details |
| 7 | Payment.cs | Model holds the Payment schema details |
| 8 | HotelContext.cs | Model holds the database context details |
| 9 | IStaff.cs | Interface for the Staff details. |
| 10 | IInventory.cs | Interface for the Inventory details. |
| 11 | IGuest.cs | Interface for the Guest details. |
| 12 | IReservation.cs | Interface for the Reservation details. |
| 13 | IRoom.cs | Interface for the Room details. |
| 14 | IRate.cs | Interface for the Rate details. |
| 15 | IPayment.cs | Interface for the Payment details. |
| 16 | StaffRepo.cs | Repository to define the Staff interface functions. |
| 17 | InventoryRepo.cs | Repository to define the Inventory interface functions. |
| 18 | GuestRepo.cs | Repository to define the Guest interface functions. |
| 19 | ReservationRepo.cs | Repository to define the Reservation interface functions. |
| 20 | RoomRepo.cs | Repository to define the Room interface functions. |
| 21 | RateRepo.cs | Repository to define the Rate interface functions. |
| 22 | PaymentRepo.cs | Repository to define the Payment interface functions. |

# Data model/Table

|  |  |  |
| --- | --- | --- |
| LOGIN | | |
|  | Username | STRING |
|  | Password | STRING |
|  | Role | STRING |

|  |  |  |
| --- | --- | --- |
| RESERVATION | | |
| PK | Id | INT |
|  | no\_of\_children | INT |
|  | no\_of\_adults | INT |
|  | no\_of\_rooms | INT |
|  | checkin\_date | DATE |
|  | checkout\_date | DATE |
|  | no\_of\_rooms | INT |
|  | PhnNumber | STRING |

|  |  |  |
| --- | --- | --- |
| GUESTS | | |
| PK | Guest\_Id | INT |
|  | Name | STRING |
|  | Address | STRING |
|  | PhnNumber | STRING |
|  | gender | STRING |

|  |  |  |
| --- | --- | --- |
| INVENTORY | | |
| PK | Id | INT |
|  | InventoryName | STRING |
|  | Quantity | INT |
|  | Price | DOUBLE |

|  |  |  |
| --- | --- | --- |
| STAFF | | |
| PK | StaffId | INT |
|  | StaffName | STRING |
|  | Address | STRING |
|  | NIC | STRING |
|  | Salary | DOUBLE |
|  | Age | INT |
|  | Occupation | STRING |
|  | Email | STRING |

|  |  |  |
| --- | --- | --- |
| ROOMS | | |
| PK | room\_id | INT |
|  | room\_type | STRING |
|  | check\_in | DATE |
|  | check\_out | DATE |
|  | status | VARCHAR(50) |

|  |  |  |
| --- | --- | --- |
| PAYMENT | | |
| PK | PaymentId | INT |
|  | CardholderName | INT |
|  | CardNumber | NVARCHAR (16) |
|  | Email | EMAIL |

|  |  |  |
| --- | --- | --- |
| RATE | | |
| PK | rate\_id | INT |
|  | No\_of\_Days | INT |
|  | ExtensionPrice | DOUBLE |
|  | PerNightPrice | DOUBLE |
|  | TotalAmount | DOUBLE |
| FK | room\_id | INT |

# 10.0 API Canvas

**Guest**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /guest | POST | Guest registration | Owner | True |
| Hotel Management System | /guest | GET | To get guest list | Owner | True |
| Hotel Management System | /guest | PUT | To update guest details | Owner | True |
| Hotel Management System | /guest /id | GET | To get details of a particular guest | Owner | True |
| Hotel Management System | /guest /id | DELETE | To remove the guest | Owner | True |

**Rooms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /room | POST | Adding rooms | Owner | True |
| Hotel Management System | /room | GET | To get room list | Owner | True |
| Hotel Management System | /room | PUT | To update room details | Owner | True |
| Hotel Management System | /room/id | GET | To get details of a particular room | Owner | True |
| Hotel Management System | /room/id | DELETE | To remove the room | Owner | True |

**Staffs**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /staff | POST | Adding staff | Owner | True |
| Hotel Management System | /staff | GET | To get staff list | Owner | True |
| Hotel Management System | /staff | PUT | To update staff details | Owner | True |
| Hotel Management System | /staff/id | GET | To get details of a particular staff | Owner | True |
| Hotel Management System | /staff/id | DELETE | To remove the staff | Owner | True |

**Inventory**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /inventory | POST | Adding inventory | Owner | True |
| Hotel Management System | /inventory | GET | To get inventory list | Owner | True |
| Hotel Management System | /inventory | PUT | To update inventory details | Owner | True |
| Hotel Management System | /inventory/id | GET | To get details of a particular inventory | Owner | True |
| Hotel Management System | /inventory/id | DELETE | To remove the inventory | Owner | True |

**Reservation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /reservation | POST | Adding reservation | Owner | True |
| Hotel Management System | /reservation | GET | To get reservation list | Owner | True |
| Hotel Management System | /reservation | PUT | To update reservation details | Owner | True |
| Hotel Management System | /reservation/id | GET | To get details of a particular reservation | Owner | True |
| Hotel Management System | /reservation/id | DELETE | To remove the reservation | Owner | True |

**Rate**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /rate | POST | Adding rate | Owner | True |
| Hotel Management System | /rate | GET | To get rate list | Owner | True |
| Hotel Management System | /rate | PUT | To update rate details | Owner | True |
| Hotel Management System | /rate/id | GET | To get details of a particular rate | Owner | True |
| Hotel Management System | /rate/id | DELETE | To remove the rate | Owner | True |

**Payment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel Management System | /payment | POST | Adding payment | Owner | True |
| Hotel Management System | /payment | GET | To get payment list | Owner | True |
| Hotel Management System | /payment | PUT | To update payment details | Owner | True |
| Hotel Management System | /payment/id | GET | To get details of a particular payment | Owner | True |
| Hotel Management System | /payment/id | DELETE | To remove the payment | Owner | True |

# 11.0 HTTP Status Code

201 – Customer Registered

200 - Request succeeded

400 – Inputs are invalid

404 – Customer Not found

502 – Bad gateway