**Online Hotel Management System**

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

DOCUMENT APPROVAL

**Approvers of this document**

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

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version #** | **Author** | **Date** | **Description** |
| 1.0 | Dipon Biswas | 07/07/2022 | Online Hotel Management System LLD |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1.0 Document Purpose 4](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636300)

[2.0 Intended Audience 4](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636301)

[3.0 Project Background, Objective(s) 4](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636302)

[3.1 Project Background 4](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636303)

[3.2 Project Objective 4](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636304)

[4.0 Design Pattern 5](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636305)

[5.0 Solution Diagram 5](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636306)

[6.0 Solution Steps 5](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636307)

[7.0 Classes/function name 6](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636308)

[8.0 Validations 6](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_Toc94636309)

[9.0 Data model/Tables 6](file:///F:\L&amp;D\Customer-management-microservice_LLD%20(1).docx#_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 Online Hotel Management Microservice.

# Intended Audience

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

|  |  |
| --- | --- |
| Role | Nature of Engagement in Online Hotel Mangement 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 microservice is a system developed to automate the major operations of the hotel where each end users can register themselves and perform various operations.

## Project Objective

The objectives of the automated 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.

# 

# Design Pattern

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | API | Using HTTP requests, we will use the respective action to trigger various operations |
| 2 | Angular | To create the client-side view of the web application |
| 3 | [SQL Server Management Studio](https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms) | Stores the various tables inside the database. |

**Hardware and Software Requirements**

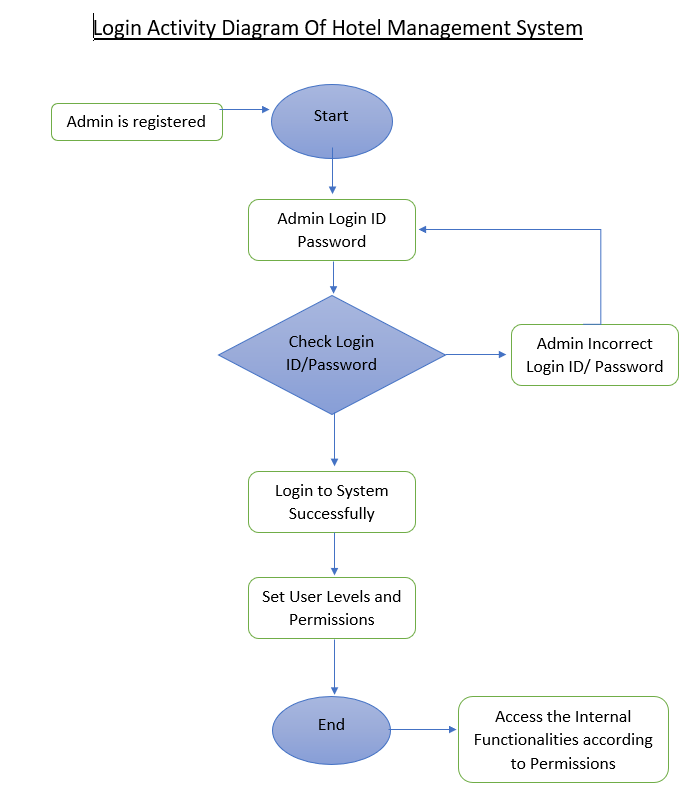
**Hardware Requirements**

|  |  |
| --- | --- |
| Processor | Core i3 or above |
| Primary Memory | 8GB |
| Hard Disk | 60GB free space |

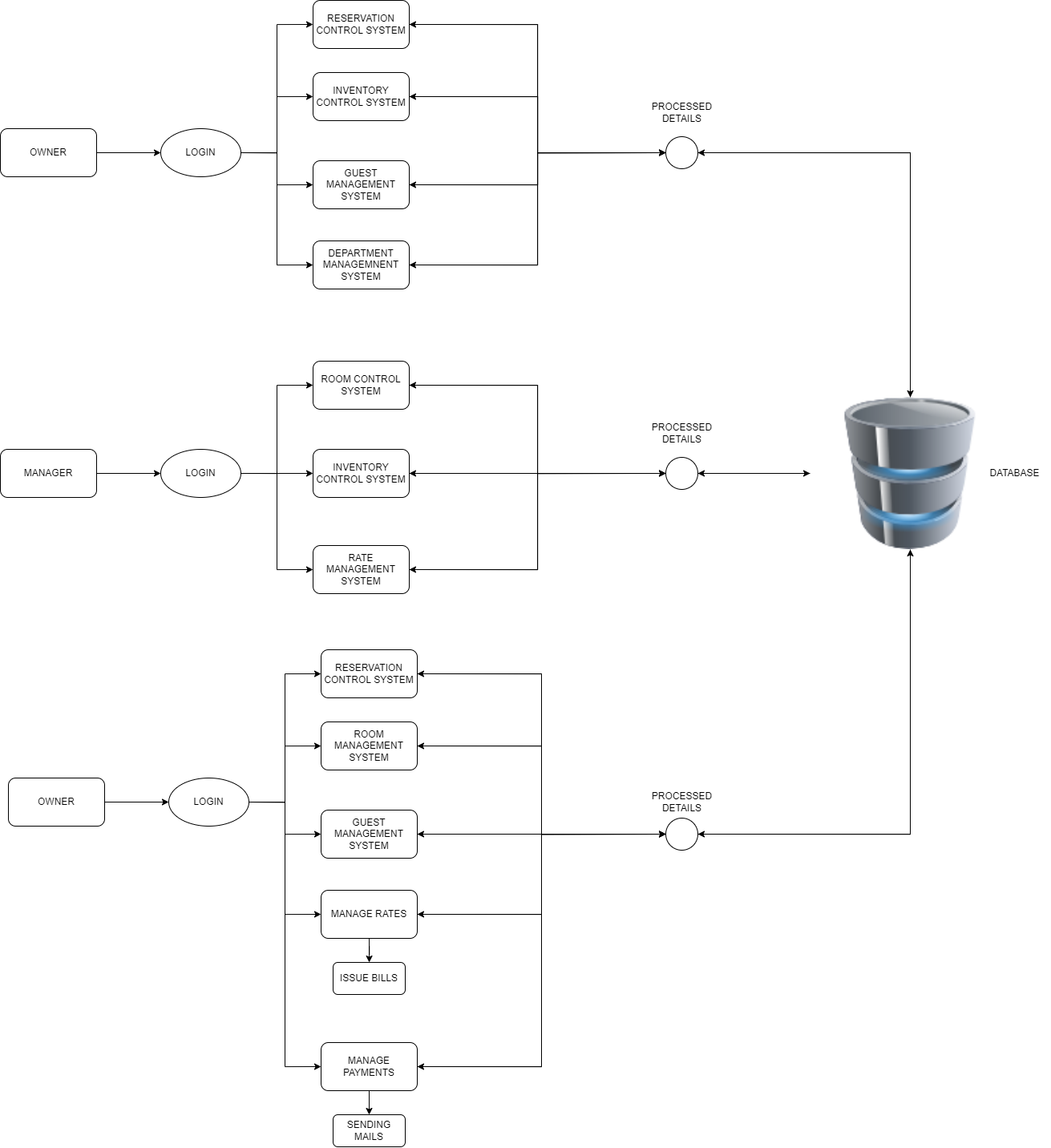
**Software Requirements**

|  |  |
| --- | --- |
| 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 |

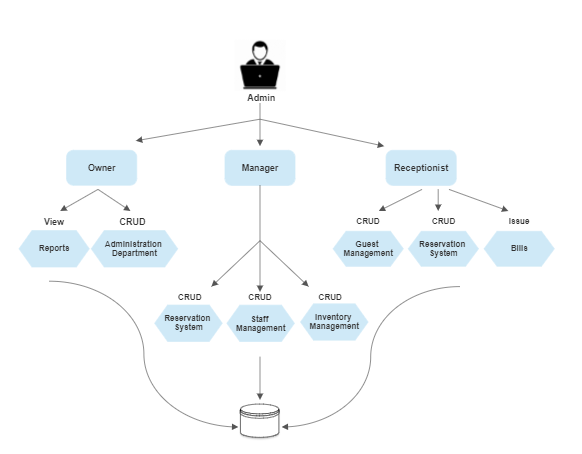
# Solution Diagram



Work Flow



Use Case Diagram

****

# 6.0 Solution Steps

**Admin Login**

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

**Search Rooms**

1. The receptionist must enter details such as 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.
3. Based on the search details provided, the receptionist can make reservation for the customers.

**Rooms**

1. Manager will enter the required room details such as room ID, check-in, check-out date, guest code and click submit button after which the browser directs the request to add the specified room
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 room
4. API gateway does the routing to check if the room is available or not.
5. Manager clicks on the Add button to add a new room details in the list as the request will pass to the addRoom() function.
6. Manager clicks on the Update button to update a room details in the list as the request will pass to the updateRoom() function.
7. Manager clicks on the Delete button to delete a room details from the list as the request will pass to the deleteRoom() function.
8. When the database is updated, the manager will be notified that the room details is added/updated/deleted successfully.
9. Manager can also view a room details by RoomDetail button which will pass the request to getRoomDetail() function.

**Make Reservation**

1. Receptionist will enter the required customer details such as 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.
   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 such as 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 according to the conditions
   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. Receptionist clicks on the Add button to add a new guest details in the list as the request will pass to the addGuest() function.
6. Receptionist clicks on the Update button to update a guest details in the list as the request will pass to the updateGuest() function.
7. Receptionist clicks on the Delete button to delete a guest details from the list as the request will pass to the deleteGuest() function.
8. When the database is updated, the receptionist will be notified that the guest is added/updated/deleted successfully.
9. Receptionist can also view a guest details by GuestDetail button which will pass the request to getGuestDetail() function.

**Staff Details**

1. Manager will enter the required guest details such as 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 according to the conditions
   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. Manager clicks on the Add button to add a new employee details in the list as the request will pass to the insertStaff() function.
6. Manager clicks on the Update button to update a Staff details in the list as the request will pass to the updateStaff () function.
7. Manager clicks on the Delete button to delete a Staff details from the list as the request will pass to the deleteStaff () function.
8. When the database is updated, the manager will be notified that the Staff is added/updated/deleted successfully.
9. Manager can also view a staff details by StaffDetail button which will pass the request to getStaffDetail() function.

**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.

**Issue Bill**

1. In order to issue bill for the customer, the receptionist must check the rate details for that particular room to get the total payment and click generate bill button.
2. The total amount will be displayed in the bill.
3. If any error arises then appropriate message will be displayed and browser redirects to pdf generation page.

# 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. |

# Table/Data Model

|  |  |  |
| --- | --- | --- |
| 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 |

# API Canvas

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Micro Service | Path | Verb | API Description | Role | Auth |
| Hotel-Management-System | /guest | POST | Guest registration | Guest | True |
| Hotel-Management-System | /guest/id | GET | To get guest list | Guest | True |
| Hotel-Management-System | /guest | PUT | To update guest details | Guest | True |
| Hotel-Management-System | /guest/id | DELETE | To remove the guest | Guest | True |
| Hotel-Management-System | /staff | POST | Staff registration | Staff | True |
| Hotel-Management-System | /staff/id | GET | To get staff list | Staff | True |
| Hotel-Management-System | /staff | PUT | To update staff details | Staff | True |
| Hotel-Management-System | /staff/id | DELETE | To remove the staff | Staff | True |
| Hotel-Management-System | /inventory | POST | Inventory registration | Inventory | True |
| Hotel-Management-System | /inventory/id | GET | To get inventory list | Inventory | True |
| Hotel-Management-System | /inventory | PUT | To update inventory details | Inventory | True |
| Hotel-Management-System | /inventory/id | DELETE | To remove the inventory | Inventory | True |
| Hotel-Management-System | /room | POST | Room registration | Room | True |
| Hotel-Management-System | /room/id | GET | To get room list | Room | True |
| Hotel-Management-System | /room | PUT | To update room details | Room | True |
| Hotel-Management-System | /room/id | DELETE | To remove room | Room | True |

# 14.0 HTTP Status Code

201 – Customer Registered

200 - Request succeeded

400 – Bad Request

401 - Unauthorized

404 – Customer Not found

502 – Bad gateway