Group No 1

Yax Avaiya

Harshana

Mankirat

Juan Pablo

HoTEL MANAGEMENT SYSTEM

Contents

[Executive Summary 2](#_Toc208063189)

[Design Analysis Process 2](#_Toc208063190)

[Actor 2](#_Toc208063191)

[Use Cases 3](#_Toc208063192)

[Use Case Diagram 4](#_Toc208063193)

[User Stories 5](#_Toc208063194)

[Functional Requirements 5](#_Toc208063195)

[Non-Functional Requirements 6](#_Toc208063196)

[Evidence of brainstorming 6](#_Toc208063197)

[Entity Relationship Diagram 8](#_Toc208063198)

[Table Designs – Date Dictionary 10](#_Toc208063199)

[Software development life cycle 14](#_Toc208063200)

[Contributions 15](#_Toc208063201)

# Executive Summary

The Hotel management System is a centralised software solution that is designed to handle key operations of a hotel in one place. It will allow staff to manage room bookings which will be both online and in-person. It will record guest details like their name, identity verification documents and their address. Processes like check-ins and check-outs, bill-handling and payments, staff and inventory monitoring will be tracked through this system.

The main purpose of this system is to reduce manual work, minimise booking errors, accelerate guest services, and allow managers to make better decisions. Overall, it will improve the work efficiency by reducing paperwork.

# Design Analysis Process

## Actors

Actors refer to the different users or roles that interact with the system. Below is a list of common actors along with brief descriptions for each:

**1. Guest / Customer**

* **Description**: A person who books a room or service at the hotel.
* **Responsibilities**:
  + Search for hotel rooms
  + Make reservations
  + Register/login (for online booking)
  + Make payments
  + Cancel bookings
  + Submit reviews or feedback

**2. Receptionist**

* **Description**: Front-desk staff responsible for handling check-in, check-out, and guest queries.
* **Responsibilities**:
  + Register guests manually
  + Assign rooms
  + Generate invoices
  + Handle walk-in customers
  + Manage check-in/check-out process
  + Provide information and support to guests

Top of Form

Bottom of Form

**3. Hotel Manager**

* **Description**: Oversees hotel operations and manages staff and customer satisfaction.
* **Responsibilities**:
  + Monitor room availability and booking reports
  + Manage staff roles and permissions
  + Review financial reports
  + View customer feedback
  + Manage pricing and discountsTop of Form

Bottom of Form

## Use Cases

1. **Login / Register**

**Actors:** Front Office Staff

**Description:** Staff logs into the system to access features securely.

INSERT INTO Customer (Customer\_id, Customer\_name, Customer\_email, Customer\_contact)

VALUES ('C006', 'Guest', 'guest@gmail.com', 987123456);

SELECT \*

FROM Customer

WHERE Customer\_email = 'guest@gmail.com'

AND Customer\_contact = 987123456;

1. **Check-in Guest**

**Actors:** Guest, Front Office Staff

**Description:** Staff registers the guest, verifies identity, and allocates a room.

INSERT INTO Booking (Booking\_id, Customer\_id, Staff\_id, Booking\_date)

VALUES ('B009', 'C009', 'S009', '2025-09-10');

INSERT INTO Schedule (Schedule\_id, Room\_id, Checkin\_time, Checkin\_date, Booking\_id)

VALUES (4, 109, '14:00', '2025-09-10', 'B009');

UPDATE Room

SET Room\_Status = 'Occupied'

WHERE Room\_id = 101;

1. **Check-out Guest**

**Actors:** Guest, Front Office Staff

**Description:** Staff finalizes billing, processes payment, and updates room status.

UPDATE Schedule

SET CheckOut\_time = '11:00', CheckOut\_date = '2025-09-12'

WHERE Booking\_id = 'B001';

UPDATE Room

SET Room\_Status = 'Available'

WHERE Room\_id = 101;

1. **Make Reservation**

**Actors:** Guest, Front Office Staff

**Description:** Booking details are added for guests making a reservation in advance.

INSERT INTO Booking (Booking\_id, Customer\_id, Staff\_id, Booking\_date)

VALUES ('B004', 'C004', 'S003', '2025-09-15');

SELECT \* FROM Booking;

1. **Cancel Reservation**

**Actors:** Guest, Front Office Staff

**Description:** An existing reservation is cancelled, and the room is freed up.

DELETE FROM Booking

WHERE Booking\_id = 'B004';

UPDATE Room

SET Room\_Status = 'Available'

WHERE Room\_id = 102;

1. **Search Room Availability**

**Actors:** Front Office Staff

**Description:** Available rooms are searched by date, room type, or status.

SELECT \*

FROM Room

WHERE Room\_status = 'Available'

AND Room\_type = 'Suite';

1. **Update Guest Information**

**Actors:** Front Office Staff

**Description:** Modifications are made to guest records such as contact or ID details.

UPDATE Customer

SET Customer\_contact = 987654321, Customer\_email = 'yaxsecond@gmail.com'

WHERE Customer\_id = 'C001';

SELECT \* FROM Customer WHERE Customer\_id = 'C001';

1. **Make Payment**

**Actors:** Guest, Front Office Staff

**Description:** Guests make payments for reservations, room charges, or services.

INSERT INTO Payment (Payment\_id, Staff\_id, Amount, Price)

VALUES ('P004', 'S002', 2, 240.00);

SELECT \*

FROM Payment

WHERE Payment\_id = 'P004';

**9. Extend Guest Stay**

**Actors:** Guest, Front Office Staff

**Description:** A guest requests to extend their stay, and staff updates the booking schedule with a new checkout date and time.

UPDATE Schedule

SET CheckOut\_date = '2025-09-15', CheckOut\_time = '12:00'

WHERE Booking\_id = 'B001';

UPDATE Room

SET Room\_Status = 'Occupied'

WHERE Room\_id = 101;

**10. View Booking Details**

**Actors:** Guest, Front Office Staff

**Description:** Guests or staff can view details of an existing booking.

SELECT \*

FROM Booking

WHERE Booking\_id = 'B001';

**11. Change Room**

**Actors:** Guest, Front Office Staff

**Description:** Staff changes a guest’s allocated room if requested.

UPDATE Schedule

SET Room\_id = 103

WHERE Booking\_id = 'B002';

UPDATE Room

SET Room\_status = 'Occupied'

WHERE Room\_id = 103;

UPDATE Room

SET Room\_status = 'Available'

WHERE Room\_id = 102;

**12. Report on payment**

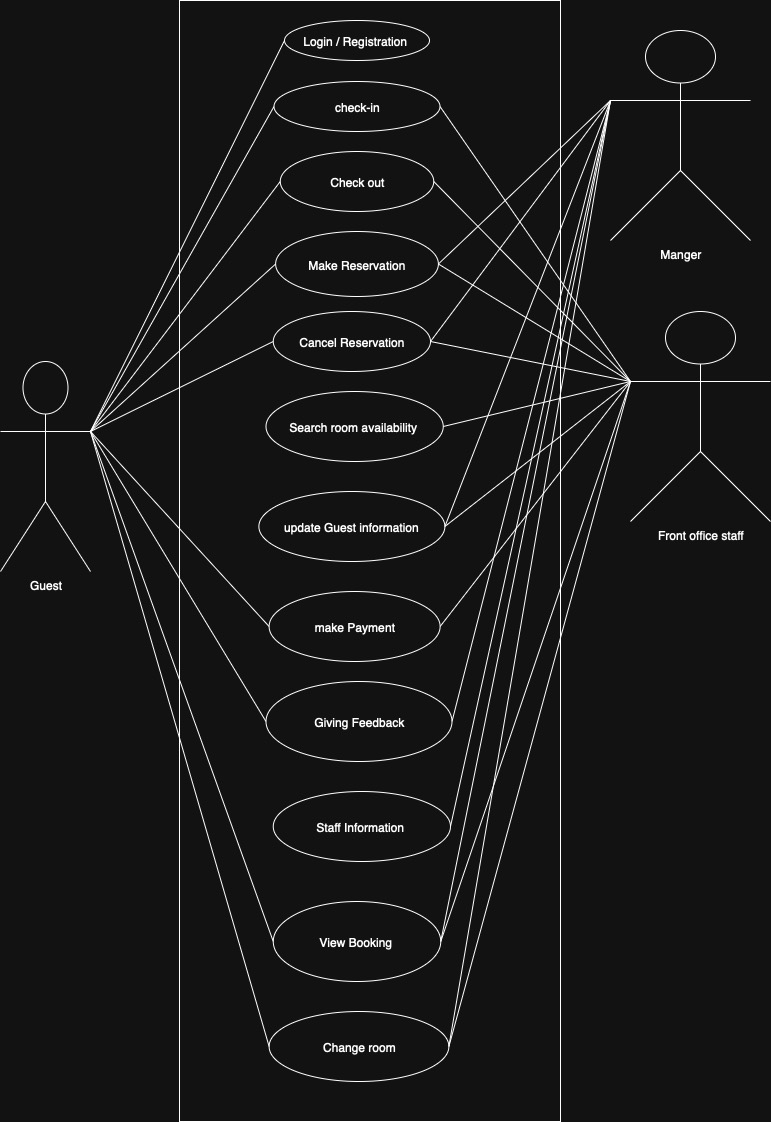
**Actors:** Front Office Staff

**Description:** Make a payment report to the manager.

SELECT Payment\_id, Staff\_id, Amount, Price

FROM Payment;

## Use Case Diagram



## User Stories

**Guest Management**

1. As a guest, I want to create an account so that I can book rooms and manage my reservations.

2. As a guest, I want to search for available rooms by date, type, and price so that I can find the best option for my stay.

3. As a guest, I want to book a room online so that I can secure my stay in advance.

4. As a guest, I want to check-in online so that I can speed up the arrival process.

5. As a guest, I want to check-out and view my bill online so that I can leave the hotel without delays.

**Room Management**

6. As a hotel staff, I want to view the room status (available, booked, under maintenance) so that I can manage room assignments effectively.

7. As a hotel staff, I want to add, edit, or remove rooms from the system so that I can keep the inventory up to date.

8. As a manager, I want to categorize rooms by type (single, double, suite) so that we can offer appropriate pricing and amenities.

**Booking & Payments**

9. As a guest, I want to pay for my reservation online using various payment methods (credit card, PayPal, etc.) so that I can confirm my booking securely.

10. As a front desk staff, I want to view all upcoming bookings so that I can prepare for guest arrivals.

**Staff Management**

11. As a manager, I want to add, update, or remove hotel staff accounts so that I can manage who has access to the system.

12. As a staff, I want to log in to the system with my credentials so that I can perform my assigned duties.

**Reports**

13. As a manager, I want to view reports on occupancy rates so that I can analyze business performance.

14. As a manager, I want to generate daily revenue reports so that I can keep track of income.

Top of Form

## Functional Requirements

Online Check-in / Pre-Arrival

* Guests must be able to complete online check-in by submitting ID and payment before arrival.
* The system should auto-assign rooms based on preferences (floor, view, bed type).

Digital Check-out

* Guests should be able to request check-out through the web portal or mobile app
* The room status must update to “Cleaning” immediately after check-out.

Reservation Management with Packages

Staff must be able to create reservations that include additional services (e.g., spa, meals, airport transfer).

* Waitlist for Fully Booked Date.
* can be added to a waitlist if no rooms are available
* Billing must allow both individual and group invoices.
* The system should notify staff and guests if a room becomes free.
* Staff must be able to search by multiple criteria (e.g., nationality, loyalty member, stay history).
* Staff should be able to mark rooms as “Available” or “Booked”.
* System must notify front desk once cleaning is complete.

Payment Management

* System must support multiple payment methods (cash, card, online wallet).
* Partial payments and deposits should be allowed and tracked.

## Non-Functional Requirements

Performance

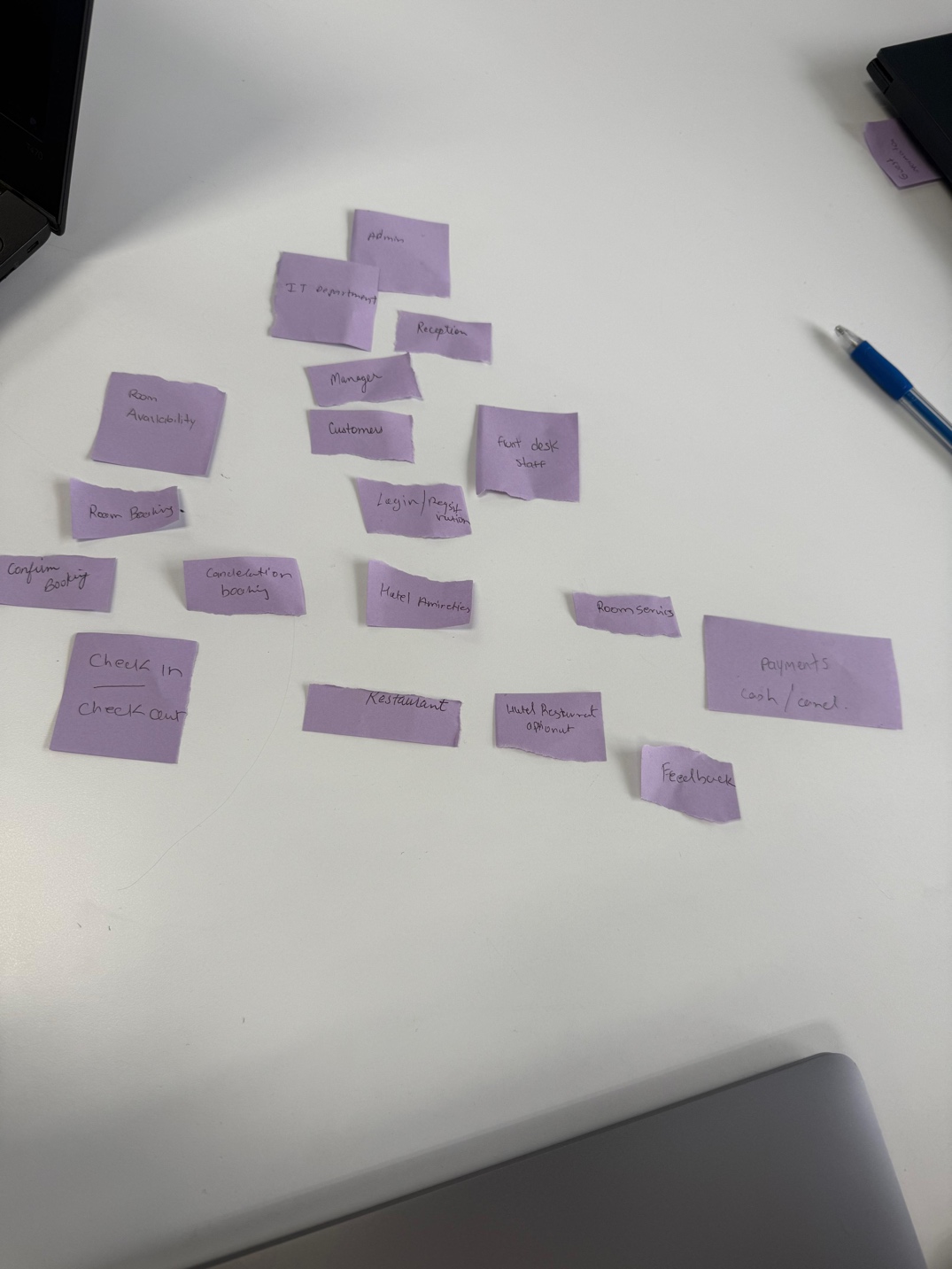
* Must process at least 100 concurrent bookings without performance issues.

**Security**

* End-to-end encryption for payment and guest data.
* Two-factor authentication (2FA) for staff login.Bottom of Form

NOTE : We will not be implementing non-functional requirements in our project.

## Evidence of brainstorming



During our initial brainstorming session (as seen in the image), we explored a wide range of hotel operations, covering various departments such as Front staff, maintenance, room, and more. This helped us understand the full scope of activities within a hotel environment.

However, after further discussion with the team, we decided to narrow the focus of our system to Front Office operations only. This shift allows us to concentrate on the core functions that directly impact guest interactions, such as reservations, check-ins, check-outs, and room assignments. By focusing on the front office.

Below we have explained each component we will be researching through:

Front Desk Operations:-

* This is the central workspace of the system, where all other modules

connect. It acts as the hotel’s digital reception:

* Overview of current check-ins, check-outs, and bookings.
* Shortcut access to billing and guest status.

Guest Booking

* Reservations: Made before arrival via phone or in-person.
* Walk-ins: Guests who arrive without a booking.
* Must include guest details, dates, room type, and status.

Internal Staff

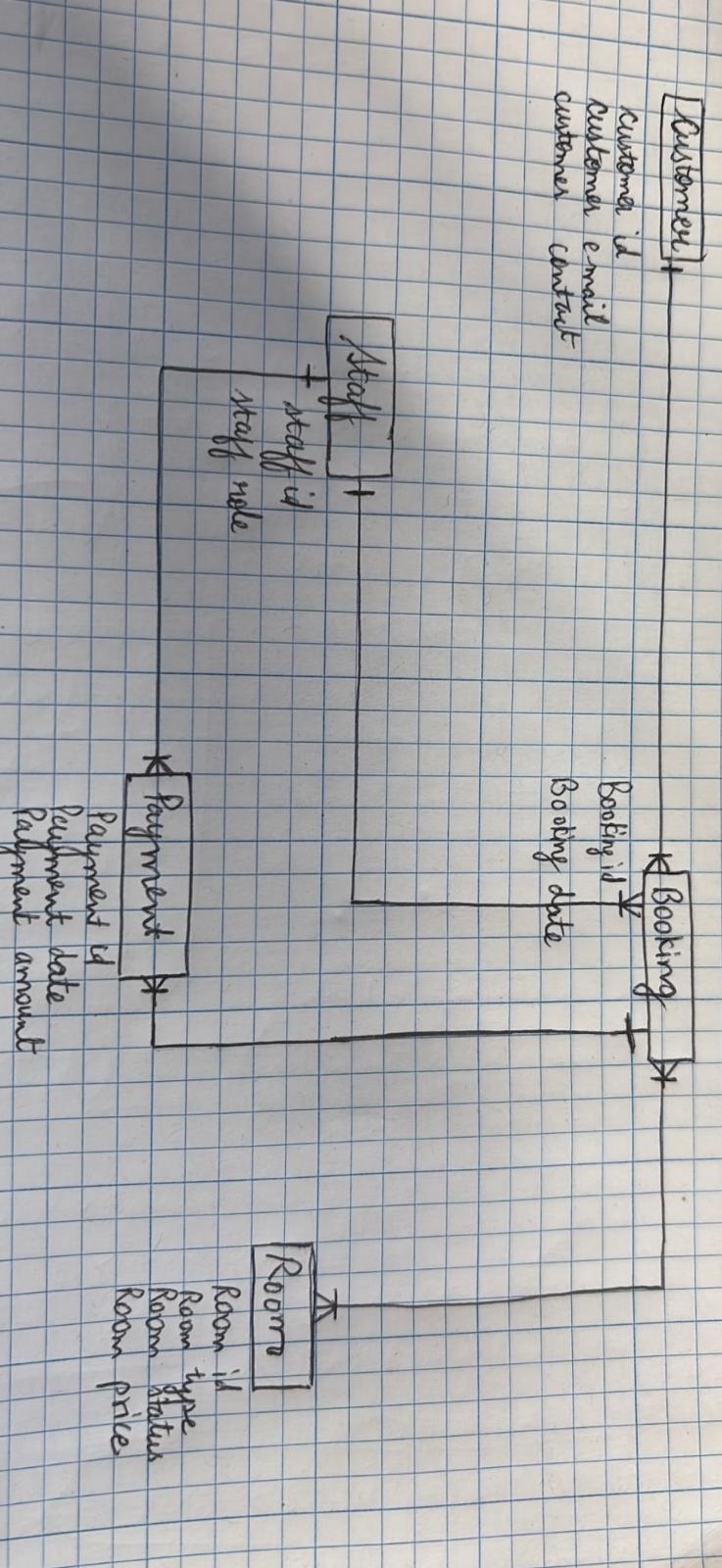
* The system will store limited internal staff data for accountability and

basic operations:

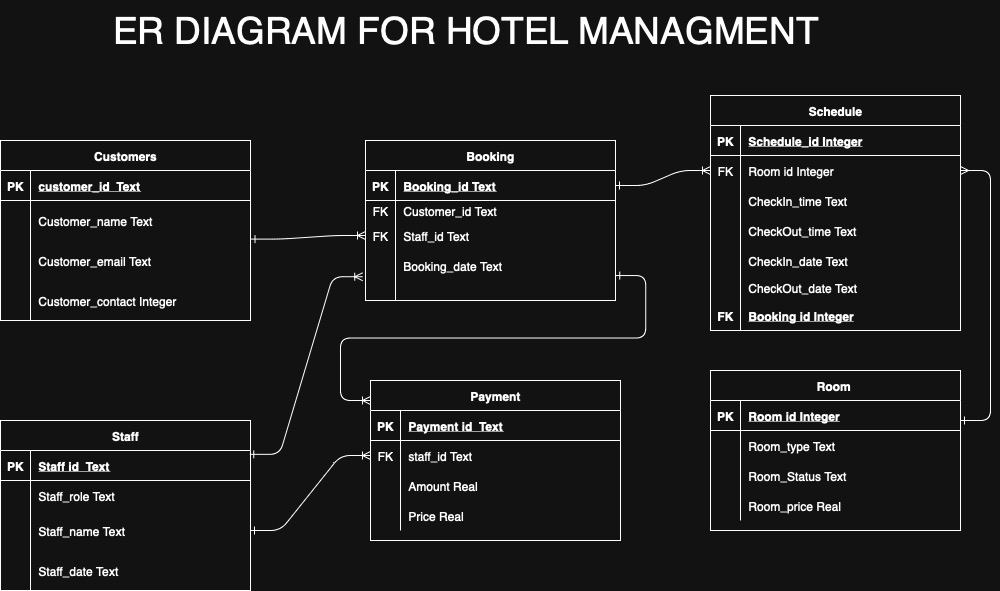
* Assign staff to check-ins/check-outs.
* Track who handled bookings or billing for auditing purposes.
* No deep HR features (e.g., payroll or shift scheduling) are included.

# Entity Relationship Diagram

**Logical ERD :**



Physical ERD:



**Relationships:-**

#### **staff–payment:**

The relationship between the staff and payment entities exhibits a one-to-many (1:N) cardinality. This means that one payment can have multiple rooms, but each payment is associated with one specific staff member.

#### **Booking–Staff:**

The Booking and staff entities are related in a one-to-many (1:N) cardinality, meaning a single Staff can Book multiple Rooms for customer, but each booking is linked to only one Staff staff member .

#### **Customer–Booking:**

The relationship between the Customer  and Booking entities exhibits a one-to-many (1:N cardinality as well. This signifies that a Customer can make multiple bookings, but each booking is linked to one specific guest.

#### **Booking–Room:**

The connection between the Booking and Rooms entities represents a Many-to-many (N:N) cardinality; a multiple booking can include multiple rooms, but each room is associated with one booking.

#### **Booking–Payment**

The relationship between the Booking and Payment entities showcases a one-to-many (1:N) cardinality. This implies that one booking can be associated with multiple payment transactions, but each payment is linked to a single booking.

# Table Designs – Date Dictionary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Customer | The customer entity includes customer personal information like name, email and contact with department information | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| Customer\_id | Unique id for each Customer | Text | PK | NOT NULL | C1, C2 |
| Customer\_name | Name of each Customer | Text |  | NOT NULL | John, Mary |
| Customer\_email | Email of each Customer | Text |  | NOT NULL | john@123.com |
| Customer\_contact | Contact number of each Customer | Integer |  | NOT NULL | 0245367891 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Staff | The staff entity includes staff personal and work information like role, name and hire date. | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| Staff \_id | Unique id for each Staff | Text | PK | NOT NULL | Staff1, Staff2 |
| Staff\_role | Role or position of staff | Text |  | NOT NULL | Desk front, Manager |
| Staff\_name | Name of each staff member | Text |  | NOT NULL | Yax, Juan |
| Staff\_date | Joining or hire date of staff | Text |  | NOT NULL | 04/09/2024 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Payment | The payment entity includes information about payments made, staff responsible, and the amounts involved | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| payment\_id | Unique id for each payment | Text | PK | NOT NULL | P100, P200 |
| Staff\_id | Staff id of the staff member who received | Text | FK | NOT NULL | Staff 1, Staff 2 |
| Amount | Amount paid | Real |  | NOT NULL | 200.50  455.90 |
| Price | Total price | Real |  | NOT NULL | $149.90, $209.99 |

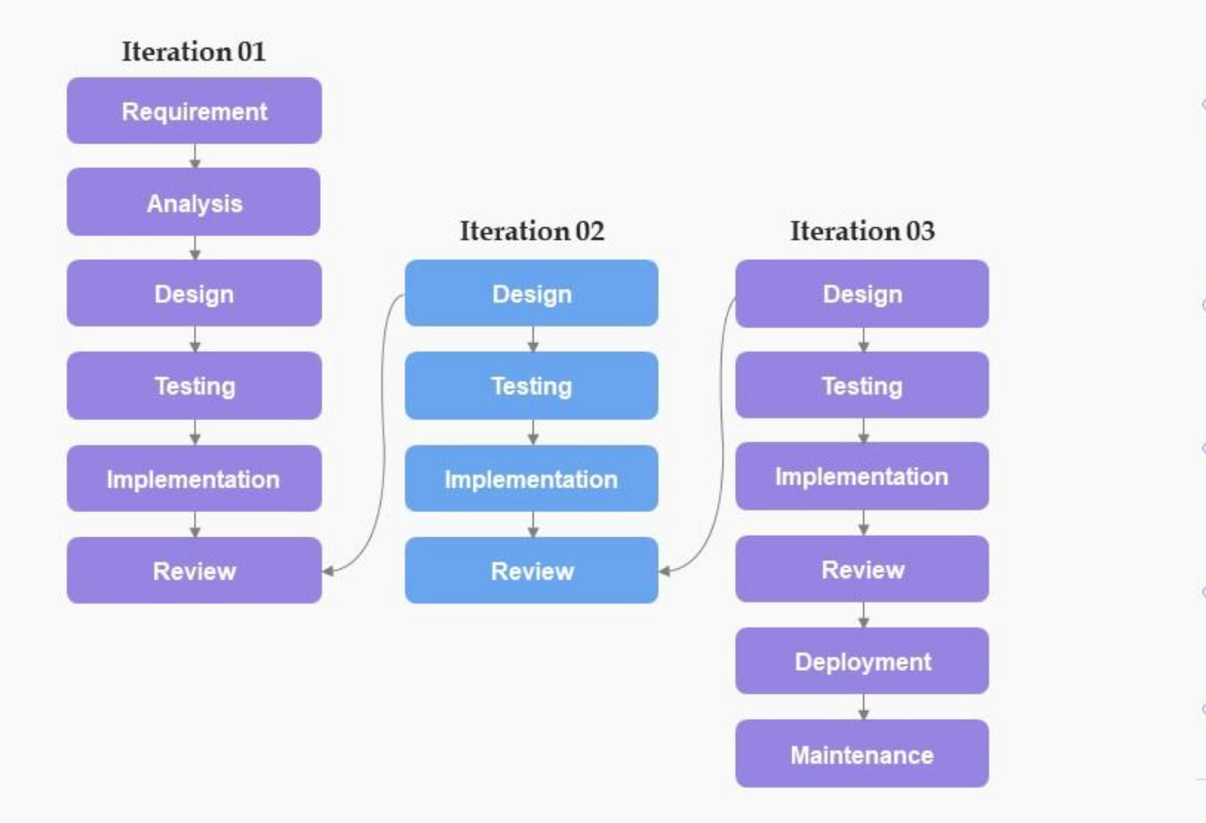
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Schedule | The schedule entity includes booking and room information, with details about check-in and check-out times and dates. | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| Booking\_id | Unique booking id from booking table | Integer | FK | NOT NULL | 0010, 0011 |
| Room\_id | Unique room id from room table | Integer | FK | NOT NULL | 311, 411 |
| CheckIn\_time | Time when guest checks in | Text |  | NOT NULL | 10:00 pm |
| CheckOut\_time | Time when guest checks out | Text |  | NOT NULL | 08:00 am |
| CheckIn\_date | Date when guest checks int | Text |  | NOT NULL | 04/09/2025 |
| CheckOut\_date | Time when guest checks out | Text |  | NOT NULL | 10/10/2025 |
| Schedule\_id | Unique Schedule id | Text |  | NOT FULL | SCH1, SCH2 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Booking | The booking entity includes booking information. | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| Booking\_id | Unique id for each booking of the room | Text | PK | NOT NULL | B1, B2 |
| Staff\_id | Unique id of each staff | Text | FK | NOT NULL | John123, Mary445 |
| Customer\_id | Unique id of each customer | Text | FK | NOT NULL | Tom45 |
| Booking\_date | Date of booking | Text |  | NOT NULL | 09/08/2025 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| entity Name | DescriptION | | | | |
| Room | The room entity includes information like room id, room type, room status and room price. | | | | |
| Field Name | **Description** | **Data type** | **Key field** | **Constraints** | **Example** |
| Room\_id | Unique id for each room | Integer | PK | NOT NULL | 311, 411 |
| Room\_type | Type of each room | Text |  | NOT NULL | Suite, Deluxe |
| Room\_status | Status of each room | Text |  | NOT NULL | Booked, Available |
| Room\_price | Price of each room | Real |  | NOT NULL | $149.90, $209.99 |

# Software development life cycle

In this project, we are going to use iterative model design of the software development life cycle, including its phases such as design & development, testing, and implementation.



The Iterative SDLC model is a software development approach where a large project is broken or divide into smaller called iterations. Instead of completing the entire project in one sequence, the software is developed into small parts and refined step by step. Each iteration includes planning, designing, coding, and testing.

This model includes feedback after every cycle to make improvements in the next stage. This model allows developers to identify and fix any problem early while they are adapting or completing the requirements.

**Why is it suitable for the project –**

The iterativemethod is quite suitable for our hotel management project because hotel systems usually need flexibility, gradual improvements, and user feedback integration. Here’s why:

1. **Progressive Development**
   * A hotel management system often has many modules: booking, check-in/check-out, billing, staff management, reports, etc.
   * With the iterative model, we can first build a small working version (e.g., room booking + billing), test it, then add more modules in later cycles.
2. **Early Functionality**
   * Instead of waiting for the whole system to be completed, the hotel can start using some features early (like reservations or guest check-ins).
   * This helps the staff adapt gradually.
3. **User Feedback**
   * Hotel staff and managers can test each version and give feedback (e.g., “billing needs GST support” or “room cleaning schedules should be linked with check-outs”).
   * Developers can then refine the system in the next iteration.
4. **Reduced Risk**
   * Errors or mismatches with real hotel operations are discovered early.
   * This prevents the cost of large-scale rework later.
5. **Flexibility for Changes**
   * Hotel policies, pricing models, (e.g., with online booking sites) may change.
   * Iterative development makes it easier to add or adjust features without redesigning the whole system.
6. **Better Testing**
   * Since each iteration produces a working version, testing can be done on real scenarios (e.g., staff handling actual guest check-ins during runs).
   * Bugs are found and fixed quickly.Top of FormBottom of Form

# Contributions

Name:- Yax Avaiya,

- Logical ER diagram

- Physical ER diagram

- Use cases diagram

- Customer Table

- SDLC model

Name:- Juan Pablo,

- Evidence from brainstorming

- Sumarry

- Staff table

- Shedule table

- SDLC model

Name:-  Mankirat kaur,

- use cases

- user stories

- actors

- Room table

- Booking table

- SDLC model

Name:- Harshana Viraj ,

- Functional requirements

- non functional requirements

- Payment Table

- SDLC model