[**Design a Hotel Management System**](https://github.com/tssovi/grokking-the-object-oriented-design-interview/blob/master/object-oriented-design-case-studies/design-a-hotel-management-system.md#design-a-hotel-management-system)

### [System Requirements](https://github.com/tssovi/grokking-the-object-oriented-design-interview/blob/master/object-oriented-design-case-studies/design-a-hotel-management-system.md#system-requirements)

We’ll focus on the following set of requirements while designing the Hotel Management System:

1. The system should support the booking of different room types like standard, deluxe, family suite, etc.
2. Guests should be able to search the room inventory and book any available room.
3. The system should be able to retrieve information, such as who booked a particular room, or what rooms were booked by a specific customer.
4. The system should allow customers to cancel their booking - and provide them with a full refund if the cancelation occurs before 24 hours of the check-in date.
5. The system should be able to send notifications whenever the booking is nearing the check-in or check-out date.
6. The system should maintain a room housekeeping log to keep track of all housekeeping tasks.
7. Any customer should be able to add room services and food items.
8. Customers can ask for different amenities.
9. The customers should be able to pay their bills through credit card, check or cash.

enum HouseKeepingStatus {

CLEAN, DIRTY,

}

Enum RoomType {

STANDARD, DELUXE, FAMILY\_SUITE, PRESIDENT;

}

Enum RoomStatus {

AVAILABLE, RESERVED, OCCUPIED, IN\_SERVICE, NOT\_AVAILABLE

}

Enum RoomBookingStatus {

REQUESTED, CANCELED, PENDING, CONFIRMED, CHECKED\_IN, CHECKED\_OUT

}

Enum BillingStatus {

PAID, WAITING\_APPROVAL, NOT\_APPROVED, PENDING, REFUNDED, REQUESTED, UN\_PAID;

}

Enum PaymentType {

CREDIT\_CARD, CASH, DEBIT\_CARD, PAY\_PAL

}

enum AccountType {

GUEST, MEMBER, RECEPTIONIST, HOUSEKEEPER, ADMIN, MANAGER

}

enum RoomServicesType {

LAUNDRY, FOOD\_OPTIONS, SPA, SAUNA, DRINKS

}

Class AdditionalServices {

Long cost;

RoomServicesType;

DateTime ordered\_DateTime;

}

Class Reservation {

String reservationId;

Room roomNo;

Account customerId;

DateTime checkInTime;

DateTime checkOutTime;

TimeStamp creationDate;

AdditionalServices additionalServices;

Payment payment;

String Reservation()

void cancelReservation(String ReservationId) {

if(TimeStamp.now()-)

}

Void addAdditionalServices() {

}

PaymentStatus initiatePayment() {

}

}

Class HouseKeeping {

Long roomNo;

HouseKeepingStatus houseKeepingStatus;

DateTime lastCleanedDate;

Account houseKeepAccount;

}

Class Room {

Long roomNo;

RoomType roomType;

Long roomCost;

HouseKeepingStatus status;

BookingStatus roomStatus;

}

Class Account {

Person person;

String accountId;

Address address;

List<Reservation> reservations;

DateTime creationDate;

AccountType accountType;

}

Class Address {

String streetAddress;

String pinCode;

String ApartmentNum;

String city;

String Country;

String County;

String State;

}

Class Person {

String personId;

String name;

String address;

String email;

String phoneNum;

String password;

Date dob;

}

Class Payment {

String paymentNum;

Reservation reservationNum;

PaymentType paymentMode;

Long totalCost;

BillingStatus statusOfPayment**;**

PaymentStatus getPaymentStatus(Reservation reservationId) {

}

}

Class NotificationService{

public NotificationService(Reservation reservationId) {

If(Time.now() – reservationId.creationDate <= 24hrs) {

sendNotif(reservation.getAccountId);

}

}

}

Class Hotel {

List<Reservation> reservationList;

Address address;

String name;

List<HouseKeeping> housekeepingStatuses;

List<Room> hotelRooms;

Reservation getReservationById(String reservationId) {

}

Reservation getReservationByAccount(Account accountID) {

}

Reservation getReservationByRoom(Room roomId) {

}

List<HouseKeeping> getHouseKeepingStatuses() {

}

List<Room> getAvailableRooms() {

}

}