

Design a Hotel Management System

Made By:

Peeyush Sinha - 1906410

Shitanshu Kumar -1906430

Tanmay Saha - 1906444

Problem Statement :

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 cancellation 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.

Use Case Diagram(UML)

-> Here are the main Actors in our system:

Guest: All guests can search the available rooms, as well as make a booking

Receptionist: Mainly responsible for adding and modifying rooms, creating room bookings, check-in, and check-out customers.

System: Mainly responsible for sending notifications for room booking, cancellation, etc.

Manager: Mainly responsible for adding new workers.

Housekeeper: To add/modify housekeeping record of rooms.

Server: To add/modify room service record of rooms.

-> Here are the top use cases of the Hotel Management System:

Add/Remove/Edit room: To add, remove, or modify a room in the system.

Search room: To search for rooms by type and availability.

Register or cancel an account: To add a new member or cancel the membership of an existing member.

Book room: To book a room.

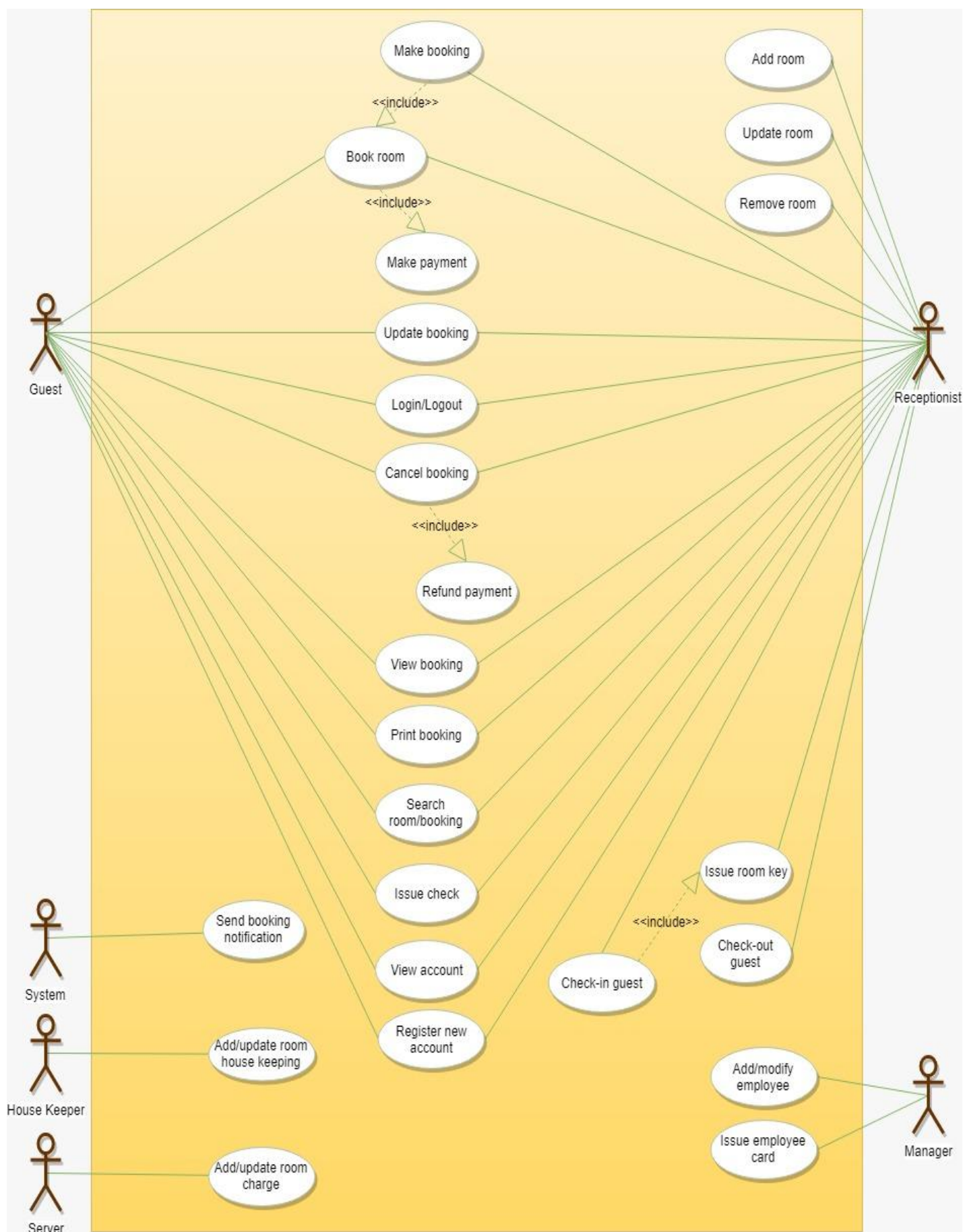
Check-in: To let the guest check-in for their booking.

Check-out: To track the end of the booking and the return of the room keys.

Add room charge: To add a room service charge to the customer's bill.

Update housekeeping log: To add or update the housekeeping entry of a room.

-> Here is the use case diagram of our Hotel Management System:

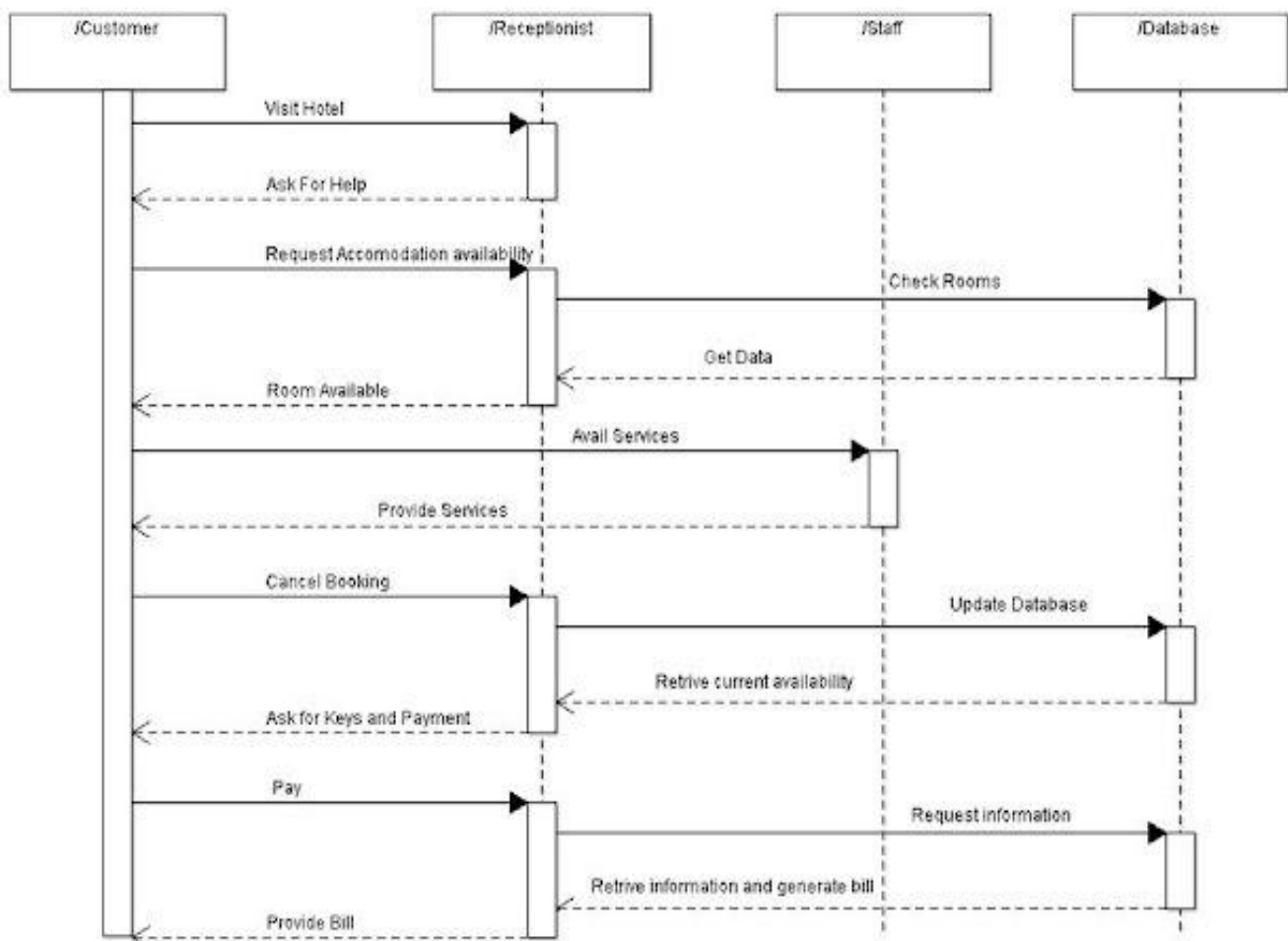


Sequence Diagram

A sequence diagram is a modeling approach for visualizing the behavioral execution of a system.

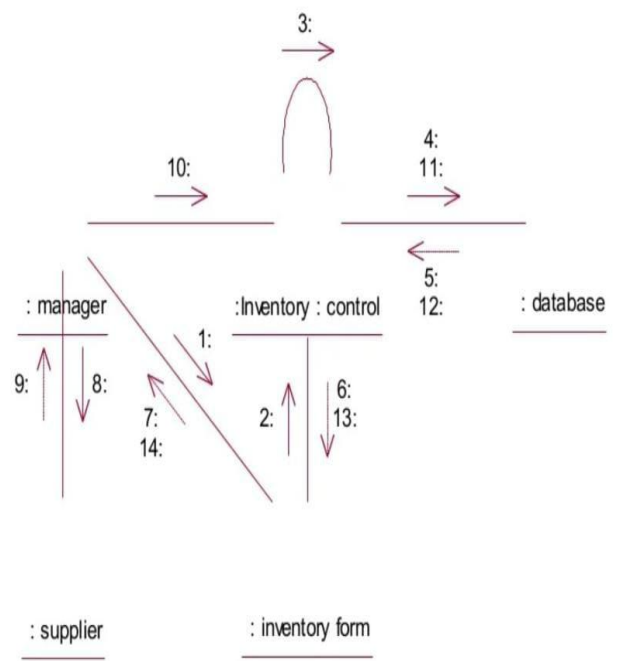
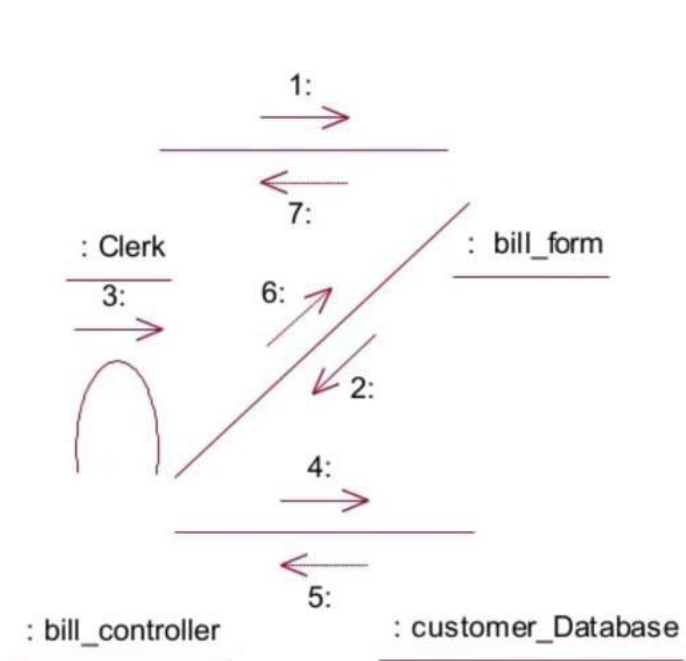
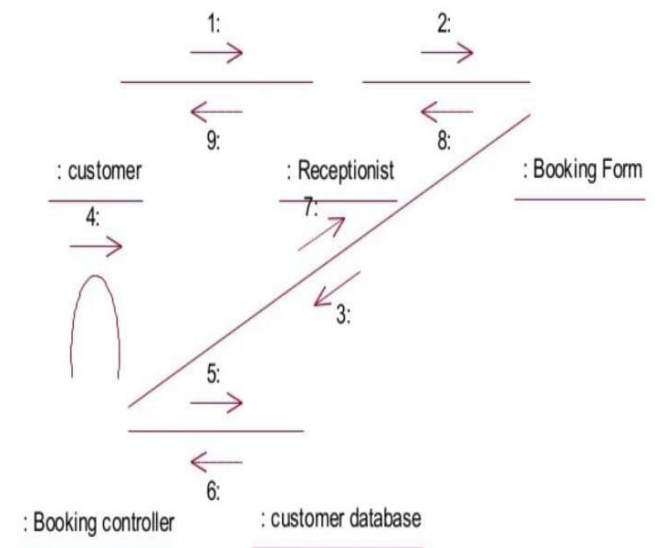
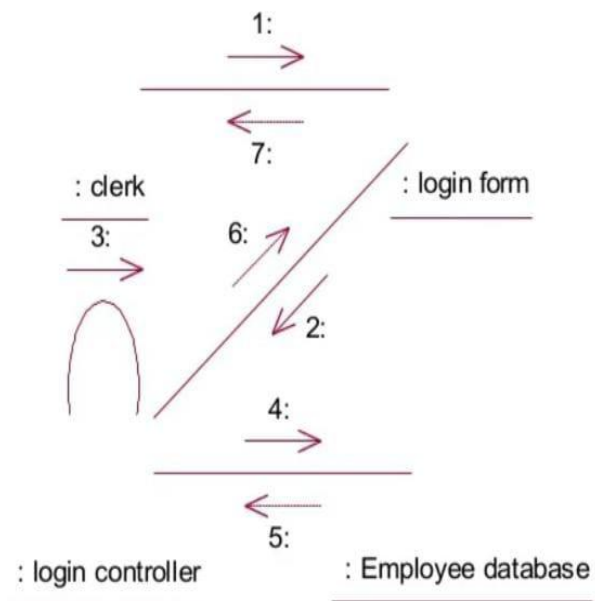
The objective of this research is to investigate the problem of security in a behavioral model (sequence diagram) through the application of model refactoring.

Diagram is as follows:-



Sequence diagram for Hotel Management System

Collaboration Diagram



Class Diagram

- > Introduction:

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity.

- > Here are the main classes of our Hotel Management System:

Hotel and Hotel Location: Our system will support multiple locations of a hotel.

Room: The basic building block of the system. Every room will be uniquely identified by the room number. Each Room will have attributes like Room Style, Booking Price, etc.

Account: We will have different types of accounts in the system: one will be a guest to search and book rooms, another will be a receptionist. Housekeeping will keep track of the housekeeping records of a room, and a Server will handle room service.

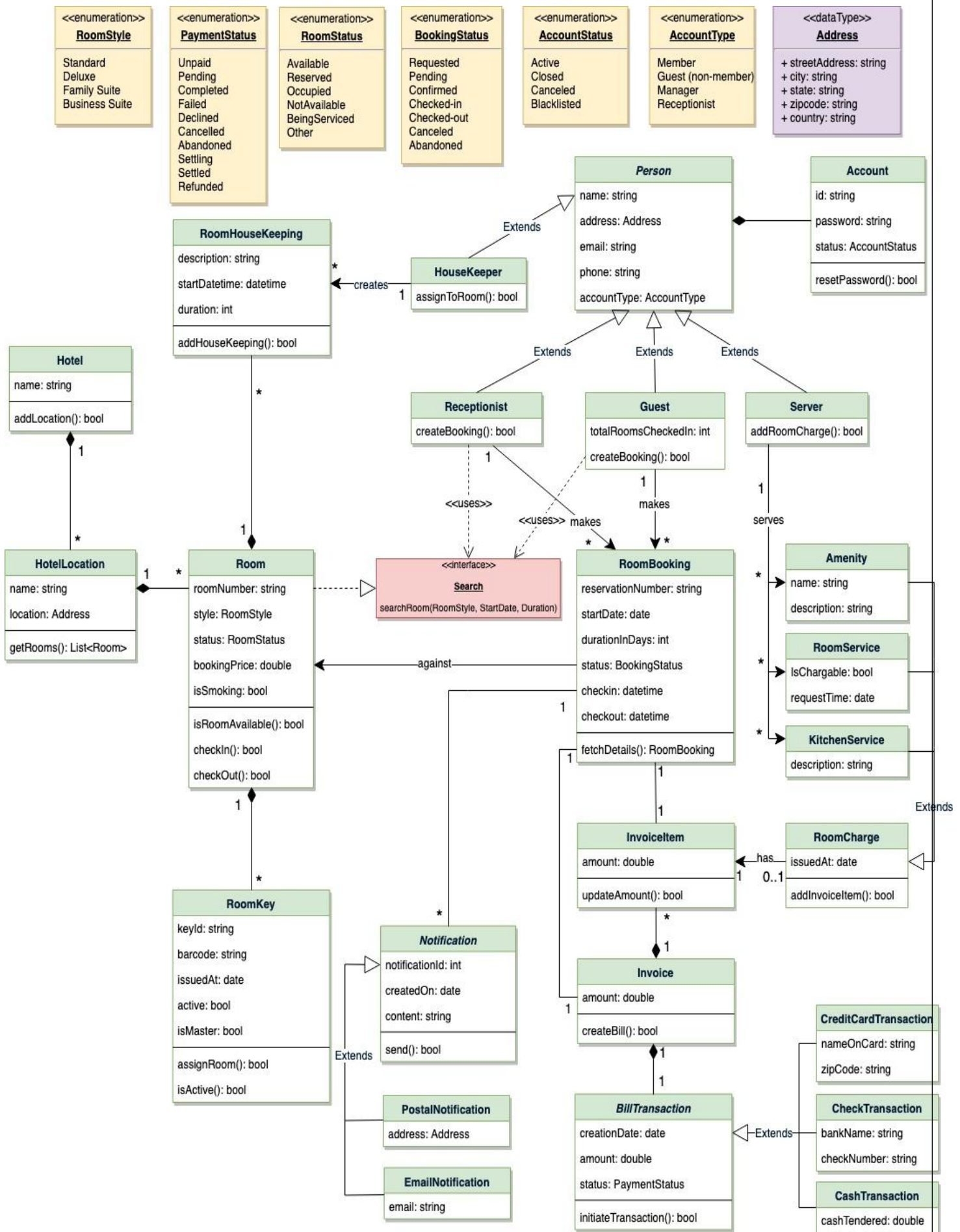
Room Booking: This class will be responsible for managing bookings for a room.
Notification: Will take care of sending notifications to guests.

Room House Keeping: To keep track of all housekeeping records for rooms.

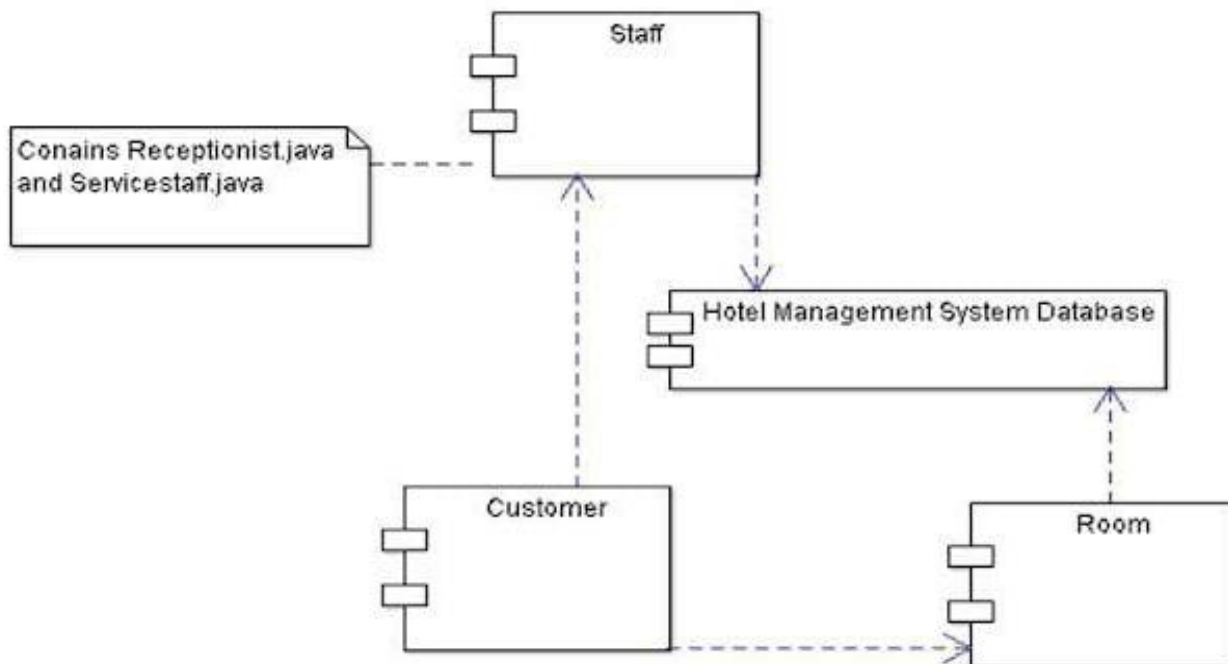
Room Charge: Encapsulates the details about different types of room services that guests have requested.

Invoice: Contains different invoice-items for every charge against the room.

Room Key: Each room can be assigned an electronic key card. Keys will have a bar-code and will be uniquely identified by a key-ID.



Object Diagram



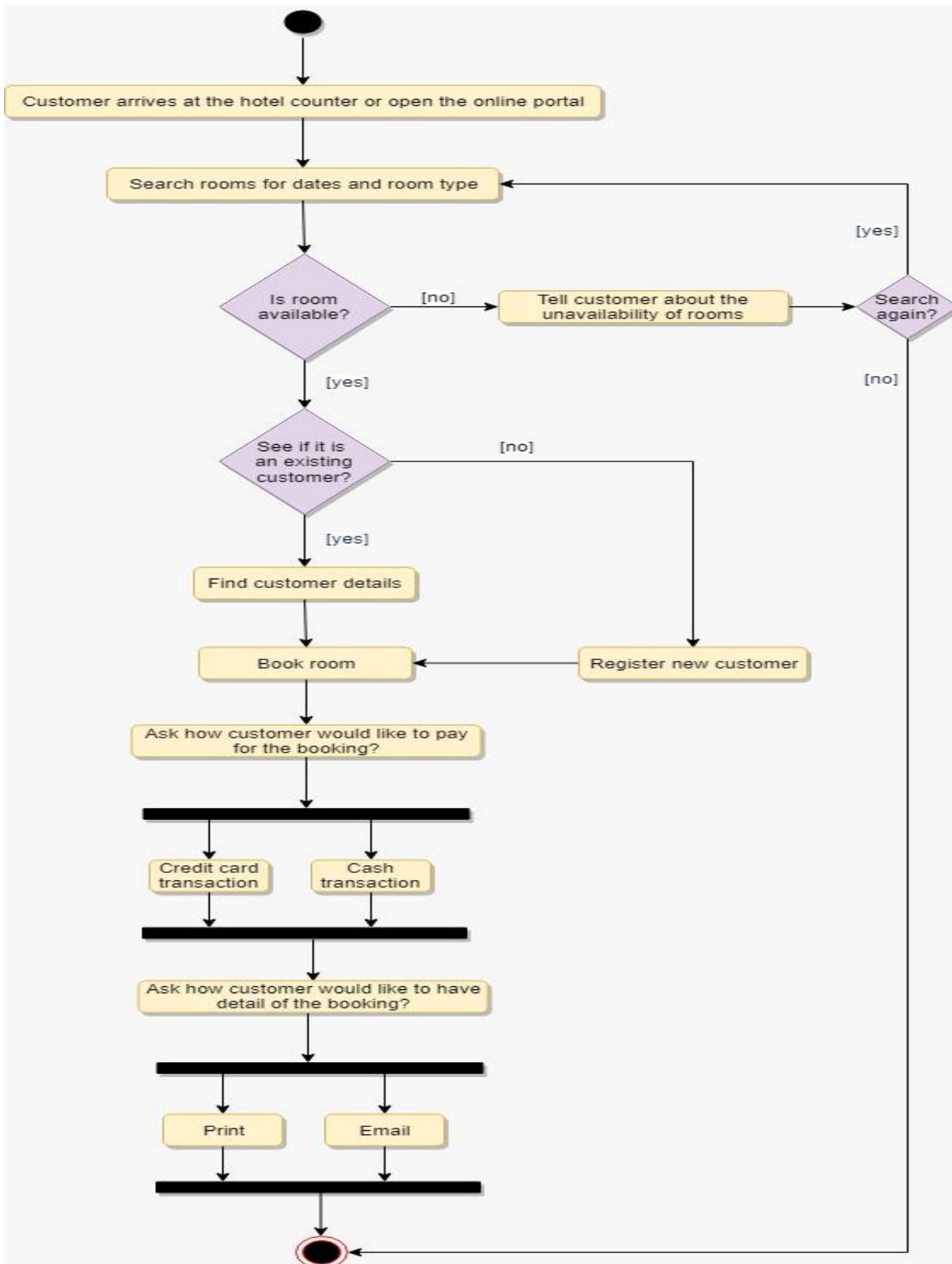
This is the UML sequence diagram of Hotel Management System which shows the interaction between the objects of Payments, Rooms, Booking, Services, Hotel.

The in stance of class objects involved in this UML Sequence Diagram of Hotel Management System are as follows:

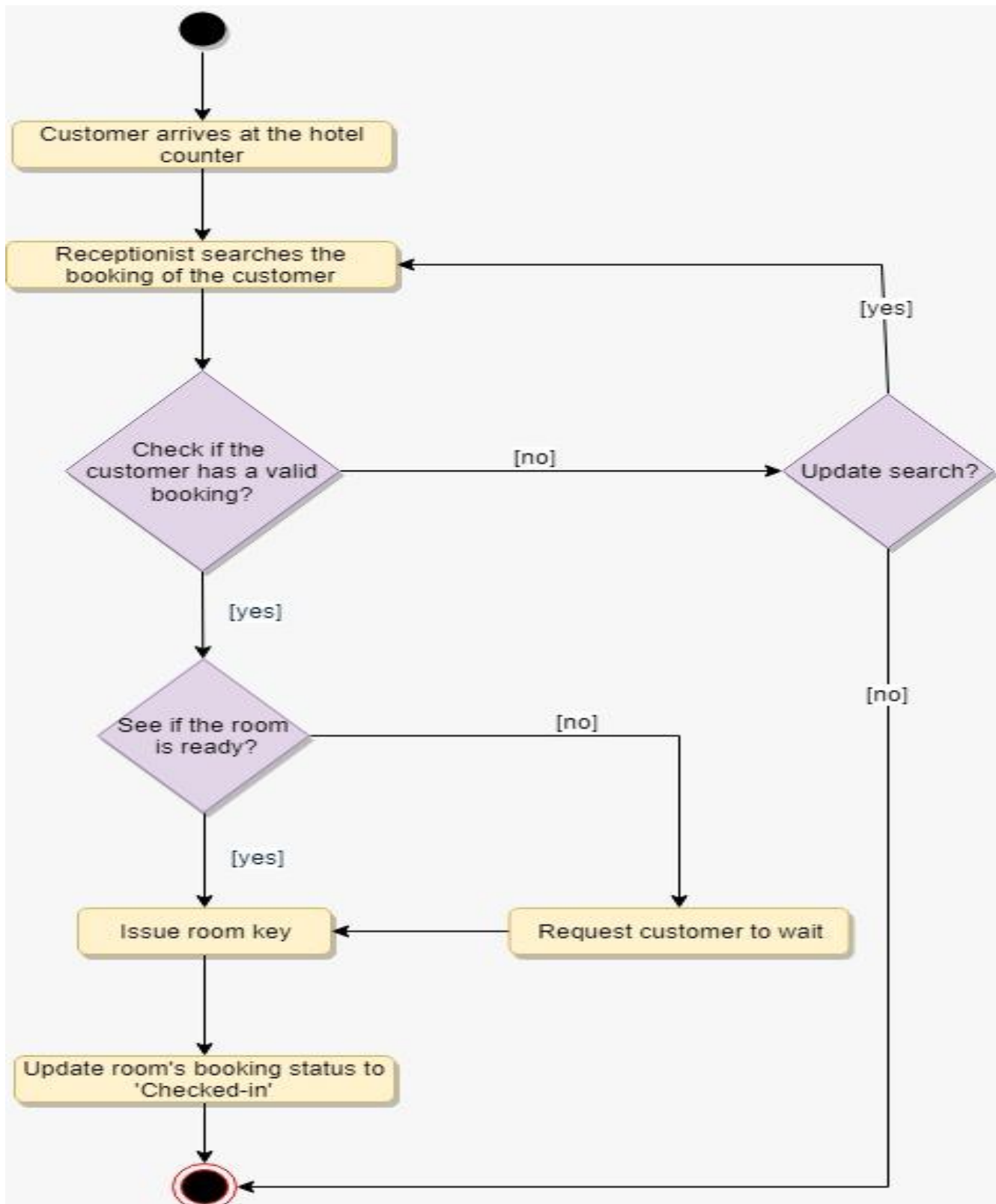
- 1) Payments Object
- 2) Rooms Object
- 3) Booking Object
- 4) Services Object
- 5) Hotel Object

Activity Diagram

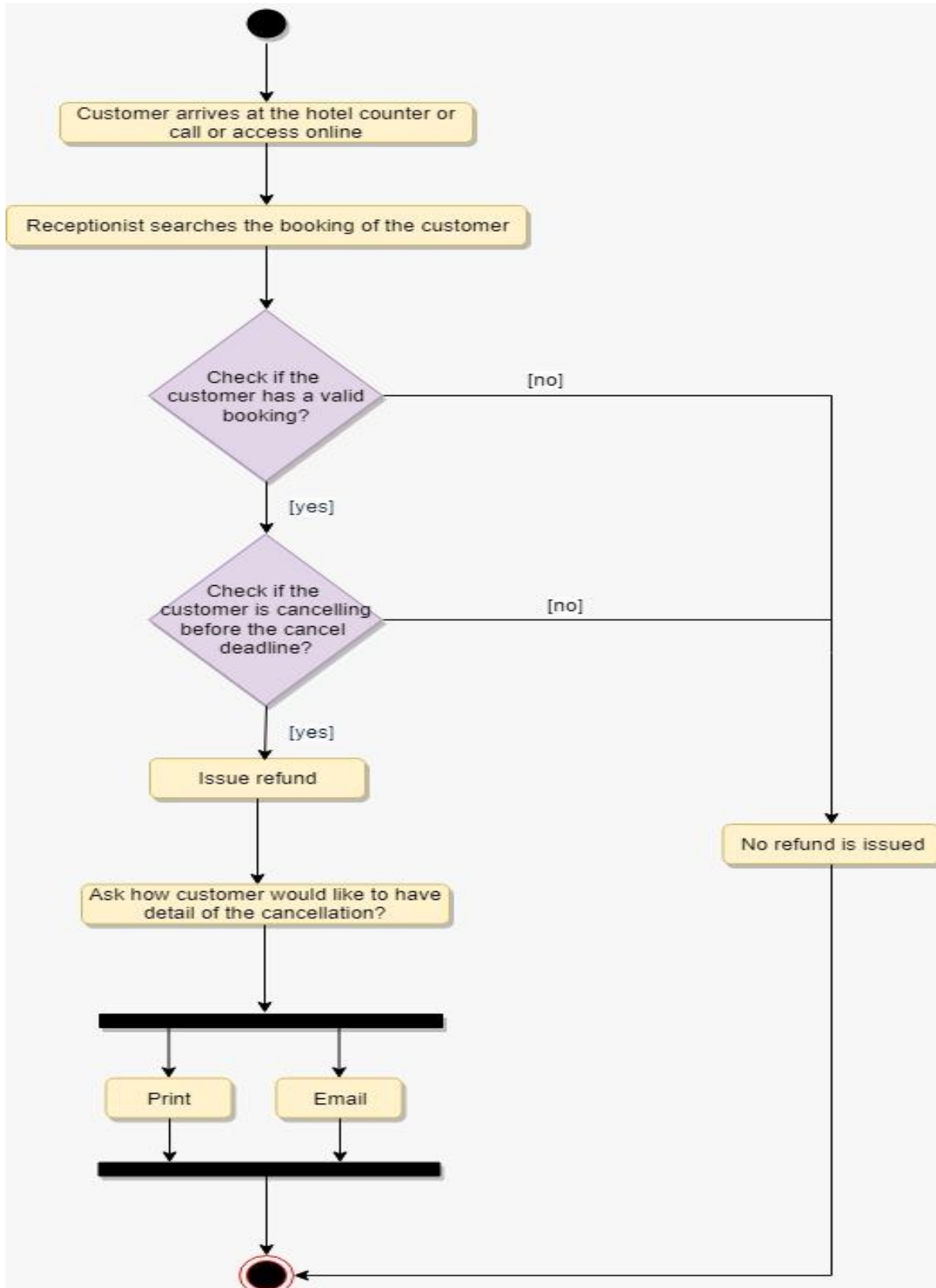
-> **Make a room booking:** Any guest or receptionist can perform this activity. Here are the set of steps to book a room :



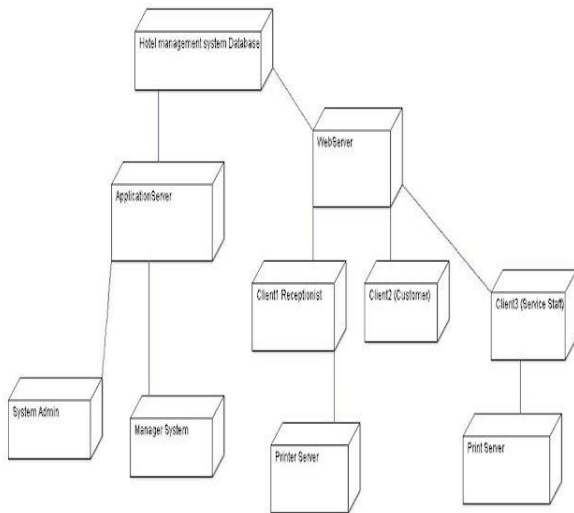
-> **Check in:** Guest will check in for their booking. The Receptionist can also perform this activity. Here are the steps:



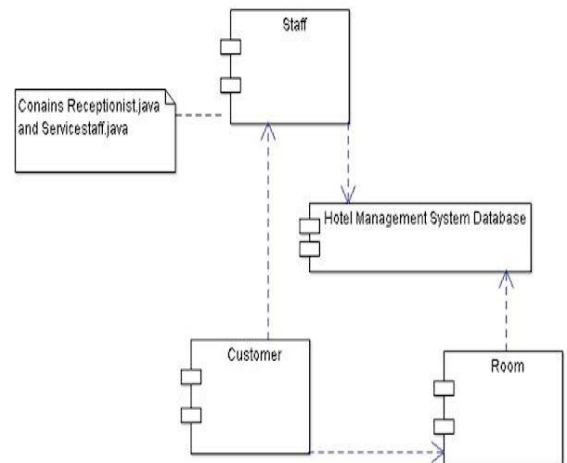
-> **Cancel a booking:** Guest can cancel their booking. Receptionist can perform this activity. Here are the different steps of this activity:



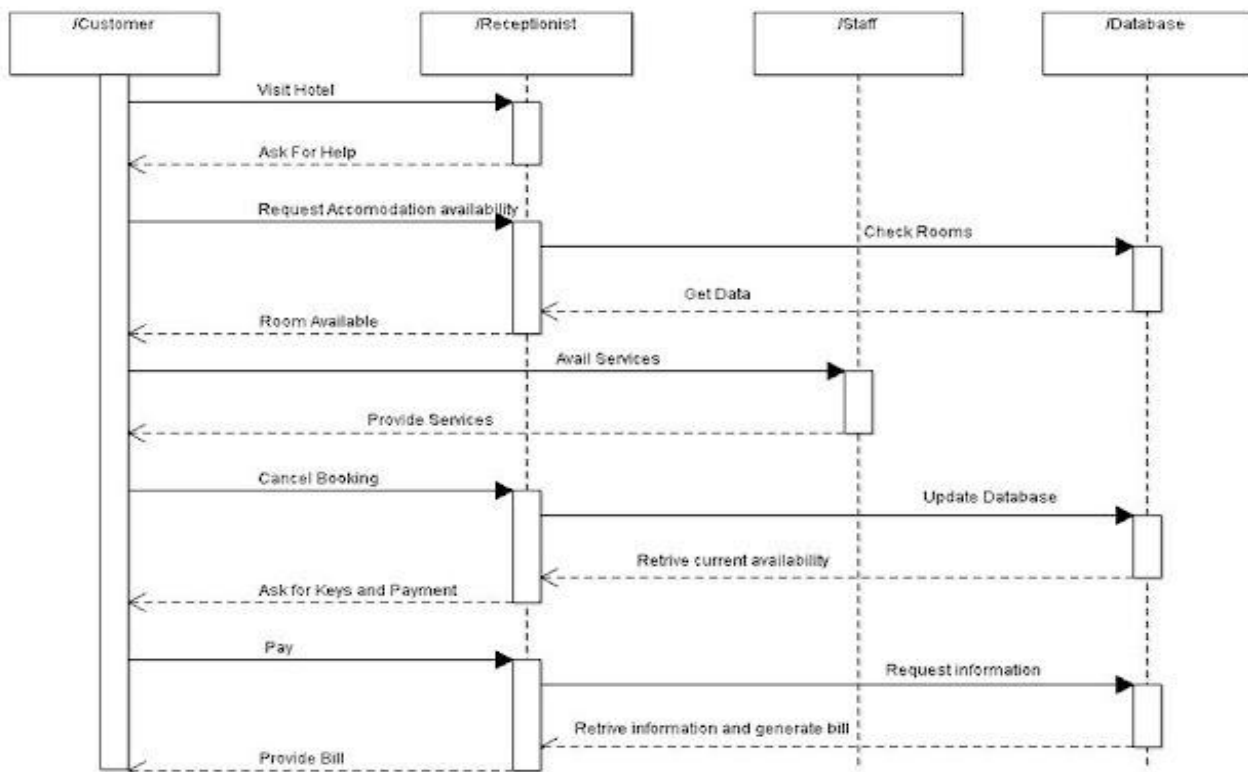
Deployment Diagram



Deployment diagram for Hotel Management System

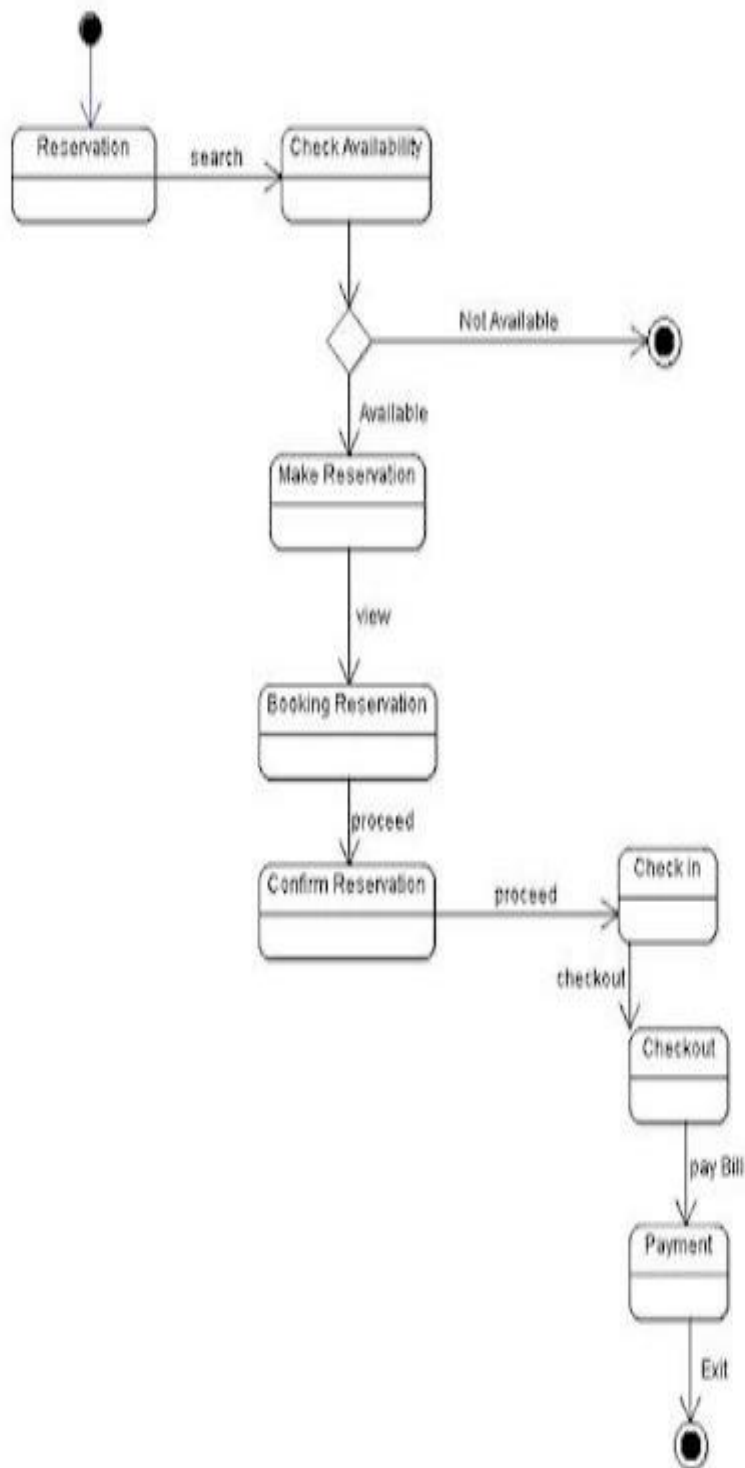


Component diagram for Hotel Management system



Sequence diagram for Hotel Management System

State - Chart Diagram



Statechart diagram for hotel management system

#Generated Code

Here is the high-level definition for the classes described above.

Enums, data types, and constants:

Here are the required enums, data types, and constants:

```
#CODE

from enum import Enum

class RoomStyle(Enum):
    STANDARD, DELUXE, FAMILY_SUITE, BUSINESS_SUITE = 1, 2, 3, 4

class RoomStatus(Enum):
    AVAILABLE, RESERVED, OCCUPIED, NOT_AVAILABLE, BEING_SERVICED, OTHER = 1, 2, 3, 4, 5, 6

class BookingStatus(Enum):
    REQUESTED, PENDING, CONFIRMED, CHECKED_IN, CHECKED_OUT, CANCELLED, ABANDONED = 1, 2, 3, 4, 5, 6, 7

class AccountStatus(Enum):
    ACTIVE, CLOSED, CANCELED, BLACKLISTED, BLOCKED = 1, 2, 3, 4, 5

class AccountType(Enum):
    MEMBER, GUEST, MANAGER, RECEPTIONIST = 1, 2, 3, 4

class PaymentStatus(Enum):
    UNPAID, PENDING, COMPLETED, FILLED, DECLINED, CANCELLED, ABANDONED, SETTLING, SETTLED, REFUNDED = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

class Address:
    def __init__(self, street, city, state, zip_code, country):
        self.__street_address = street
        self.__city = city
        self.__state = state
        self.__zip_code = zip_code
        self.__country = country
```

Account, Person, Guest, Receptionist, and Server:

These classes represent the different people that interact with our system:

```
#CODE

from abc import ABC
from .constants import *

# For simplicity, we are not defining getter and setter functions. The reader can
# assume that all class attributes are private and accessed through their respective
# public getter methods and modified only through their public methods function.

class Account:
    def __init__(self, id, password, status=AccountStatus.Active):
        self.__id = id
        self.__password = password
        self.__status = status

    def reset_password(self):
        None

# from abc import ABC, abstractmethod
class Person(ABC):
    def __init__(self, name, address, email, phone, account):
        self.__name = name
        self.__address = address
        self.__email = email
        self.__phone = phone
        self.__account = account

class Guest(Person):
    def __init__(self):
        self.__total_rooms_checked_in = 0

    def get_bookings(self):
        None

class Receptionist(Person):
    def search_member(self, name):
        None

    def create_booking(self):
        None

class Server(Person):
    def add_room_charge(self, room, room_charge):
        None
```


Hotel and Hotel Location:

These classes represent the top-level classes of the system:

```
#CODE

class HotelLocation:
    def __init__(self, name, address):
        self.__name = name
        self.__location = address

    def get_rooms(self):
        None

class Hotel:
    def __init__(self, name):
        self.__name = name
        self.__locations = []

    def add_location(self, location):
        None
```

Room, RoomKey, and RoomHouseKeeping:

To encapsulate a room, room key, and housekeeping:

```
from datetime import datetime
from abc import ABC

class Search(ABC):
    def search(self, style, start_date, duration):
        None

class Room(Search):
    def __init__(self, room_number, room_style, status, price,
is_smoking):
        self.__room_number = room_number
        self.__style = room_style
        self.__status = status
        self.__booking_price = price
        self.__is_smoking = is_smoking

        self.__keys = []
        self.__house_keeping_log = []

    def is_room_available(self):
        None

    def check_in(self):
        None

    def check_out(self):
        None

    def search(self, style, start_date, duration):
        None
```

return all rooms with the given style and availability

```
class RoomKey:
def __init__(self, key_id, barcode, is_active, is_master):
self.__key_id = key_id
self.__barcode = barcode
self.__issued_at = datetime.date.today()
self.__active = is_active
self.__is_master = is_master

def assign_room(self, room):
None

def is_active(self):
None

class RoomHouseKeeping:
def __init__(self, description, duration, house_keeper):
self.__description = description
self.__start_datetime = datetime.date.today()
self.__duration = duration
self.__house_keeper = house_keeper

def add_house_keeping(self, room):
None
```

RoomBooking and RoomCharge:

To encapsulate a booking and different charges against a booking:

```
from datetime import datetime
from abc import ABC

class RoomBooking:
    def __init__(self, reservation_number, start_date, duration_in_days,
booking_status):
        self.__reservation_number = reservation_number
        self.__start_date = start_date
        self.__duration_in_days = duration_in_days
        self.__status = booking_status
        self.__checkin = None
        self.__checkout = None

        self.__guest_id = 0
        self.__room = None
        self.__invoice = None
        self.__notifications = []

    def fetch_details(self, reservation_number):
        None

# from abc import ABC, abstractmethod
class RoomCharge(ABC):
    def __init__(self):
        self.__issue_at = datetime.date.today()

    def add_invoice_item(self, invoice):
        None

class Amenity(RoomCharge):
    def __init__(self, name, description):
        self.__name = name
        self.__description = description

class RoomService(RoomCharge):
    def __init__(self, is_chargeable, request_time):
        self.__is_chargeable = is_chargeable
        self.__request_time = request_time

class KitchenService(RoomCharge):
    def __init__(self, description):
        self.__description = description
```