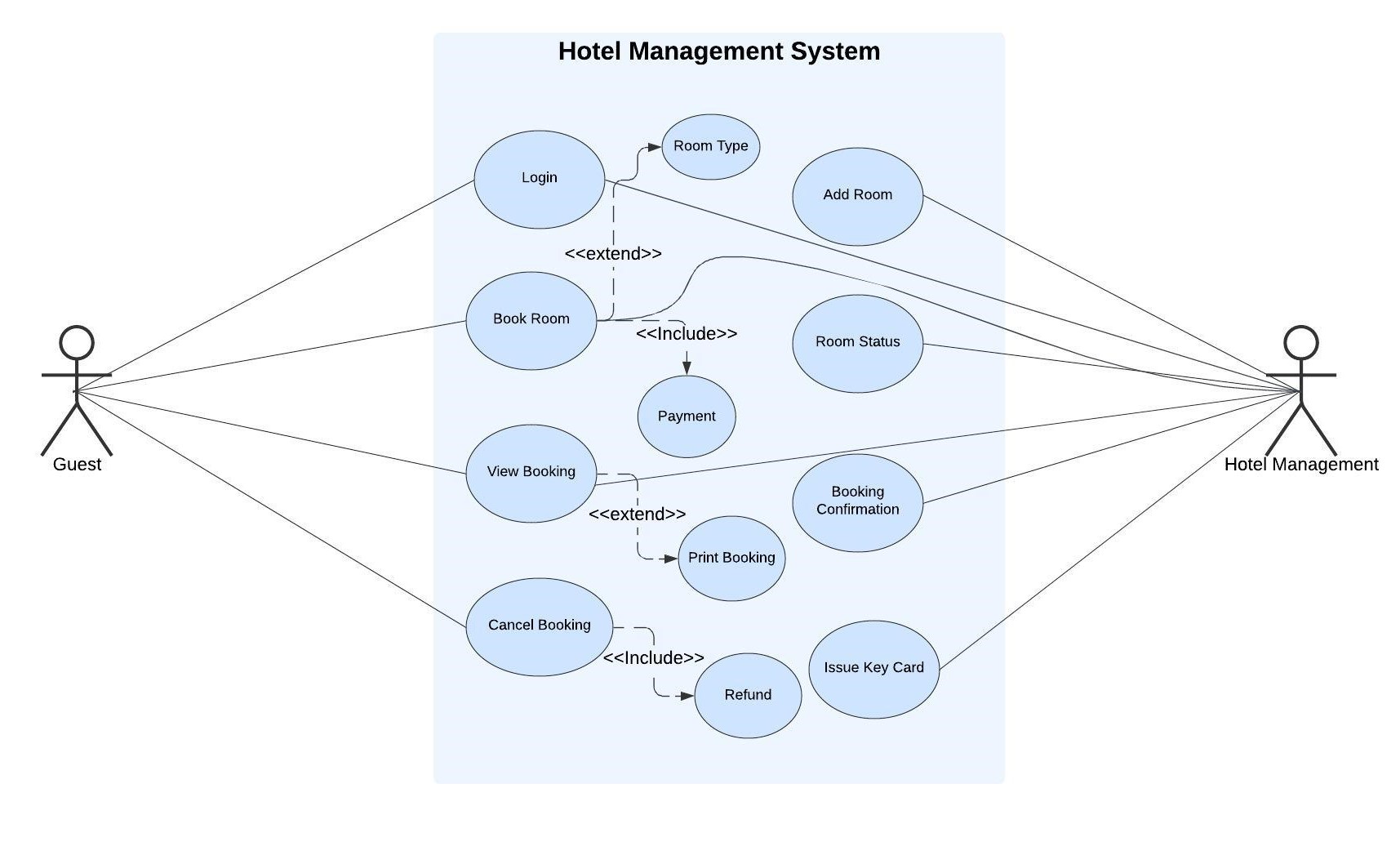
**UML Use-Case-Diagram**

****

**1. Use Case Descriptions for Guests**

**1. Login**

**Description:** Allow users to authenticate themselves to access the system.

**Actors:** Hotel staff, guests (for online booking).

**Precondition:** User has a valid username and password.

**Postcondition:** User is successfully logged into the system.

**Steps:**

1. User enters their username and password.
2. System verifies the credentials against the database.
3. If credentials are valid, the system grants access to the appropriate features.
4. If credentials are invalid, the system displays an error message.

**2. Book Room**

**Description:** Allow guests to reserve a room for a specific period.

**Actors:** Guests **Precondition:** Guest is logged in to the system.

**Postcondition:** Room is reserved for the specified period.

**Steps:**

1. Guest selects desired room type, check-in and check-out dates.
2. System checks room availability for the specified dates.
3. If room is available, system calculates the total cost.
4. Guest provides payment information.
5. System processes the payment and confirms the reservation.
6. System sends a confirmation email to the guest.

**3. View Booking**

**Description:** Allow guests and hotel staff to view existing reservations.

**Actors:** Guests, hotel staff **Precondition:** User is logged in to the system.

**Postcondition:** User can view details of their reservations.

**Steps:**

1. User selects the "View Bookings" option.
2. System displays a list of the user's reservations.
3. User can click on a reservation to view its details, such as room type, check-in/check-out dates, and payment status.

**4. Cancel Booking**

**Description:** Allow guests to cancel their existing reservations.

**Actors:** Guests **Precondition:** Guest has an active reservation.

**Postcondition:** Reservation is canceled.

**Steps:**

1. Guest selects the "Cancel Booking" option.
2. System displays a list of the user's reservations.
3. Guest selects the reservation they want to cancel.
4. System confirms the cancellation.
5. System updates the room availability and refunds the guest if applicable.
6. System sends a cancellation confirmation email to the guest.

**2. Use Case Descriptions for Hotel Management**

**1. Add Room**

**Description:** Allow receptionists to add new rooms to the hotel's inventory.

**Actors:** Receptionists **Precondition:** Receptionist is logged in to the system.

**Postcondition:** New room is added to the database.

**Steps:**

1. Receptionist selects the "Add Room" option.
2. Receptionist enters room details such as room number, type, capacity, and amenities.
3. Receptionist sets the room rate.
4. System adds the new room to the database.

**2. Room Status**

**Description:** Allow receptionists to view and update the status of rooms.

**Actors:** Receptionists **Precondition:** Receptionist is logged in to the system.

**Postcondition:** Room status is updated in the database.

**Steps:**

1. Receptionist selects the "Room Status" option.
2. System displays a list of all rooms and their current status (e.g., available, occupied, cleaning).
3. Receptionist can click on a room to view its details and update its status.
4. Possible status updates include:
   * Available: Room is ready for occupancy.
   * Occupied: Room is currently occupied by a guest.
   * Cleaning: Room is being cleaned.
   * Out of Order: Room is not available for occupancy due to maintenance or repairs.

**3. Booking Confirmation**

**Description:** Allow receptionists to confirm or reject guest bookings.

**Actors:** Receptionists

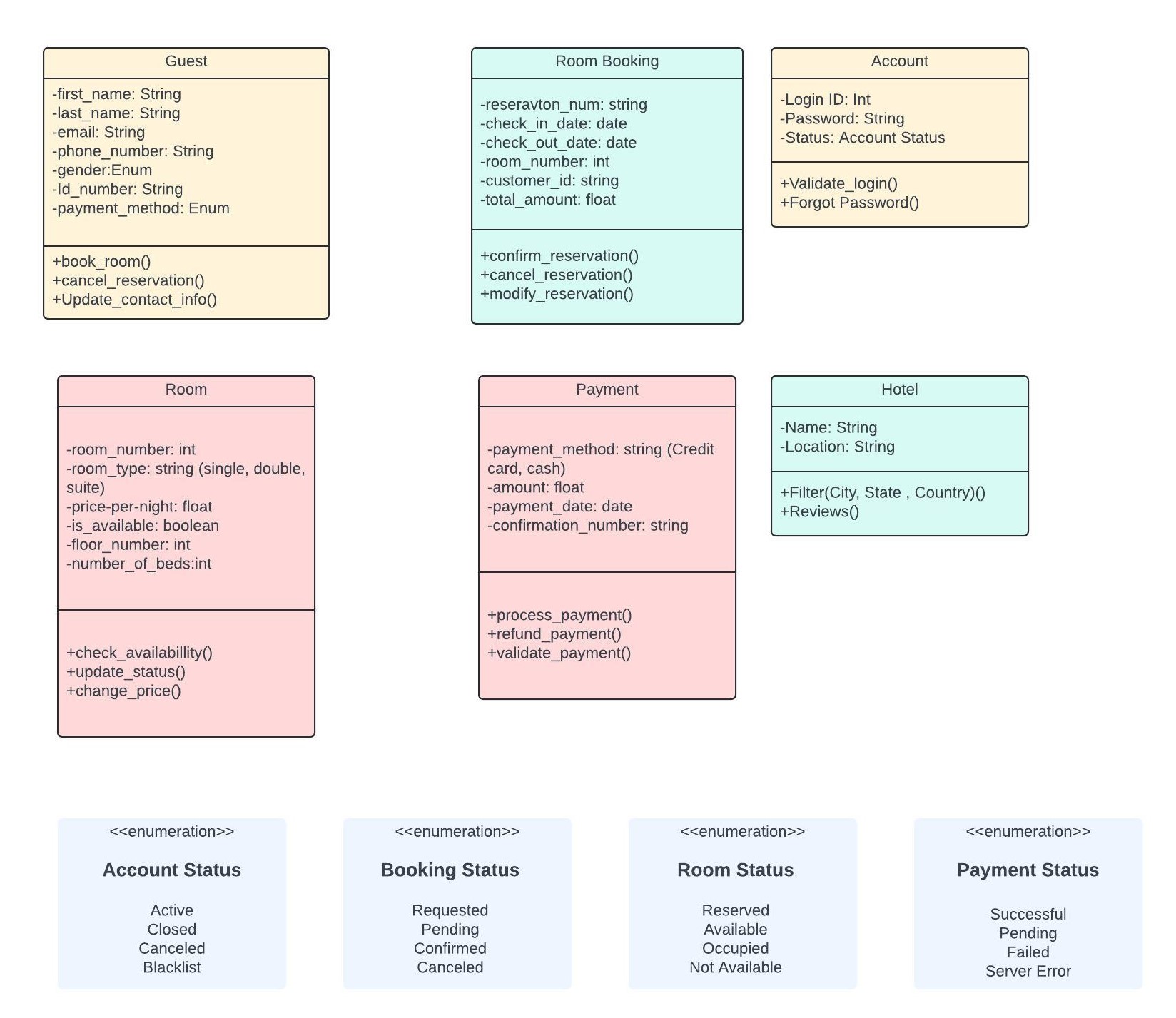
**Precondition:** Receptionist is logged in to the system and there is a pending booking request.

**Postcondition:** Booking is confirmed or rejected.

**Steps:**

1. Receptionist selects the "Booking Confirmation" option.
2. System displays a list of pending booking requests.
3. Receptionist selects a booking request to review.
4. Receptionist verifies the guest's information and room availability.
5. If the booking is valid, receptionist confirms the reservation and sends a confirmation email to the guest.
6. If the booking is invalid (e.g., room is not available), receptionist rejects the reservation and sends a rejection email to the guest.

**UML Class Diagrams**

****

***Python Classes (Code):***

class Hotel:

    def \_\_init\_\_(self, name, location):

        self.name = name

        self.location = location

        self.rooms = []

        self.employees = []  # Consider using a separate class for users

        self.bookings = []

    def add\_room(self, room):

        self.rooms.append(room)

    def remove\_room(self, room):

        self.rooms.remove(room)

    # ... other functions like view\_rooms, view\_employees, etc.

    def add\_guest(self, guest):  # New method for adding guests

        # Add logic for guest registration and validation

        self.guests.append(guest)

    def add\_account(self, account):  # New method for adding accounts (optional)

        # Add logic for account creation and validation

        self.accounts.append(account)

    def add\_booking(self, guest, room, booking):  # Corrected method

        self.bookings.append(booking)

class Guest:

    def \_\_init\_\_(self, name, address, phone\_no, id\_no, email):

        self.name = name

        self.address = address

        self.phone\_no = phone\_no

        self.id\_no = id\_no

        self.email = email

class Account:

    def \_\_init\_\_(self, id, password, account\_status):

        self.id = id

        self.password = password

        self.account\_status = account\_status

class Room:

    def \_\_init\_\_(self, room\_no, room\_type, room\_price, room\_status):

        self.room\_no = room\_no

        self.room\_type = room\_type

        self.room\_price = room\_price

        self.room\_status = room\_status

class RoomBooking:

    def \_\_init\_\_(self, booking\_id, start\_date, end\_date, days\_duration, booking\_status):

        self.booking\_id = booking\_id

        self.start\_date = start\_date

        self.end\_date = end\_date

        self.days\_duration = days\_duration

        self.booking\_status = booking\_status

    def add\_payment(self, payment):

        # Function to add a payment to the booking

        pass

class Payment:

    def \_\_init\_\_(self, amount, mode\_of\_payment, payment\_status):

        self.amount = amount

        self.mode\_of\_payment = mode\_of\_payment

        self.payment\_status = payment\_status

***Python Object Creation (Code):***

hotel = Hotel("Hotel Name", "Abudhabi")

hotel.guests = []  # Added a list for guests

hotel.accounts = []  # Added a list for accounts (optional)

guest1 = Guest("Suliman", "123 Main St", "555-1234", "123456789", "johndoe@example.com")

account1 = Account("suliman123", "password123", "active")  # (Optional)

room1 = Room("101", "Deluxe", 200.0, "available")

room\_booking1 = RoomBooking("B123", "2024-10-01", "2024-10-05", 5, "confirmed")

payment1 = Payment(1000.0, "credit card", "successful")

# Add room to the hotel

hotel.add\_room(room1)

# Add guest (implement logic in add\_guest method)

hotel.add\_guest(guest1)  # Call the new method

# Add account (optional, implement logic in add\_account method)

hotel.add\_account(account1)  # Call the new method (if used)

# Create a booking (corrected)

hotel.add\_booking(guest1, room1, room\_booking1)

# Add payment to the booking

room\_booking1.add\_payment(payment1)

# View hotel details

print(hotel.name)

print(hotel.location)

# View guest details

print(guest1.name)

print(guest1.address)

# View room details

print(room1.room\_no)

print(room1.room\_type)

# View booking details

print(room\_booking1.booking\_id)

print(room\_booking1.start\_date)

# View payment details

print(payment1.amount)

print(payment1.mode\_of\_payment)

# Print all objects

print(hotel)

print(guest1)

print(room1)

print(room\_booking1)

print(payment1)

**Github Link -** <https://github.com/ayush1713/Hotel-Management-System-UML>

***Summary of Learnings***

Working on this UML software project as a student has allowed me to develop a profound knowledge of using UML for modeling and designing software systems. I have gained understanding on generating different kinds of UML diagrams, like use case diagrams, class diagrams, and sequence diagrams, for representing the functionality and structure of a system. I have also learned important lessons about the significance of precise requirements, efficient communication, and teamwork among colleagues. This project has enhanced my problem-solving abilities and my capacity to critically analyze software design. I think the abilities I've gained from this project will be extremely useful as I further my education and seek a profession in software development.