

## **Hotel Reservation System**

Sheikha H. Almoalla

College of Interdisciplinary Studies, Zayed University

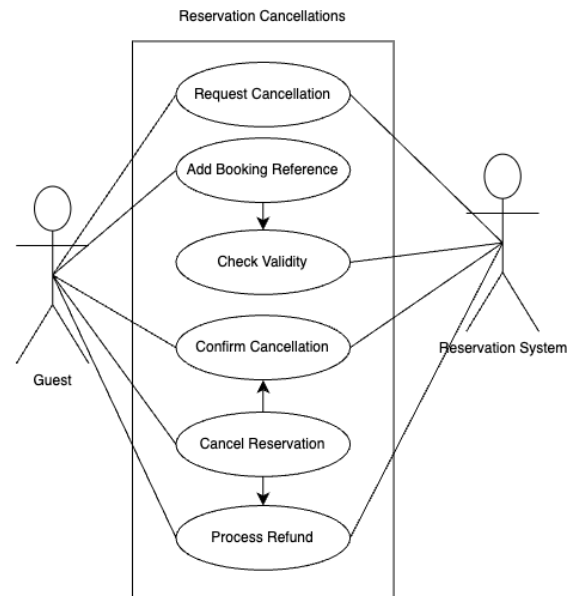
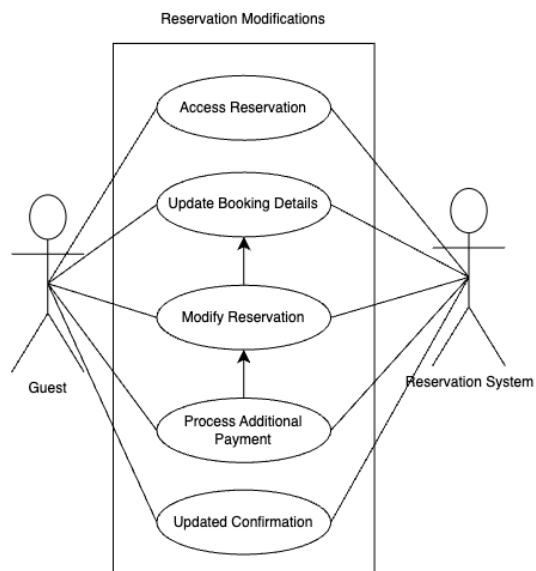
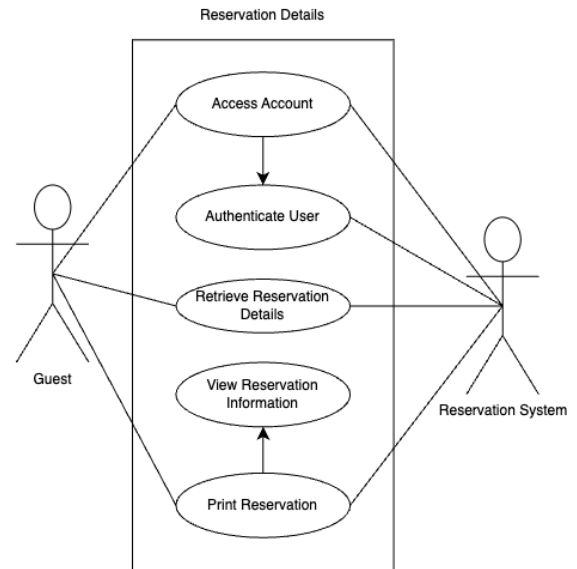
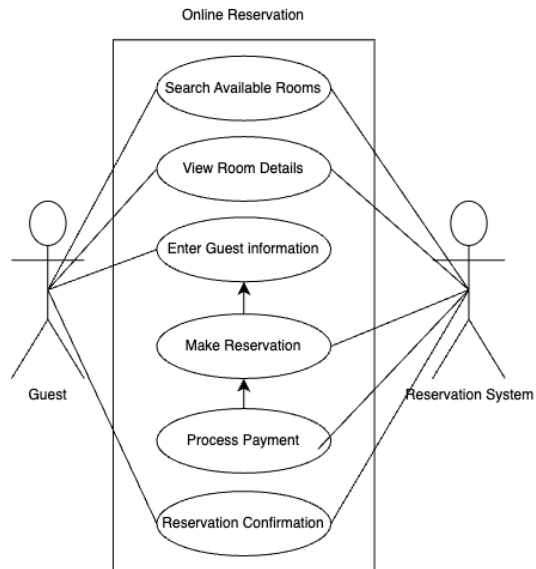
22528 Program. Fund.

Dr. Sujith Mathew

September 30, 2024

# Hotel Reservation System

## Use-Case Diagrams:



## Use-Case Descriptions:

### Scenario: Room Availability

<b>Use Case:</b>	Search Available Rooms
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	Guests want to book a room, so they start by searching for available rooms from the hotel's system
<b>Preconditions:</b>	The rooms are available on the hotel's reservation system
<b>Main Scenarios:</b>	<ol style="list-style-type: none"><li>1. Guests are searching on the online reservation system of the hotel</li><li>2. Guests would enter their criteria or preferences through the filter bar (e.g., dates, room type, number of guests).</li><li>3. The hotel's system would process the query.</li><li>4. The hotel's system would retrieve and display the available rooms based on the guest's preferences.</li></ol>
<b>Exceptions:</b>	
<b>4a.</b>	No rooms available that match the guest's preferences
	<ol style="list-style-type: none"><li>1. The system informs the guest.</li><li>2. The system suggests modifying the search criteria.</li></ol>

### Scenario: View Room Details

<b>Use Case:</b>	View Room Details
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	Guests select a specific room they are interested in from the list available and view the rooms' detailed information
<b>Preconditions:</b>	The guest has performed a room search, and room information is displayed in the hotel's system.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"><li>1. The guests select their preferred room and view the details.</li><li>2. The system retrieves the information of the room (i.e price, photos, bed size, and other amenities)</li><li>3. The system would display that to the guests.</li></ol>
<b>Exceptions:</b>	

<b>3a.</b>	The selected room is no longer available.
	<ol style="list-style-type: none"> <li>1. The system informs the guest of the misfortune.</li> <li>2. The system suggests going back and searching for the other available rooms.</li> </ol>

### Scenario: Enter Guest Information

<b>Use Case:</b>	Enter Guest Information
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	Guests found a suitable room and proceeded to check out.
<b>Preconditions:</b>	The guests have to select rooms and proceed with check-out
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guests would enter their personal information (special requests, full name, email address, and contact information).</li> <li>2. The guests would review the information.</li> <li>3. The system would validate the information by sending a confirmation email.</li> <li>4. The guest would authenticate it via email.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	Information is not valid (e.g., incorrect email address) or incomplete (e.g., missing phone number)
	<ol style="list-style-type: none"> <li>1. The system would prompt the guest to correct the information.</li> </ol>

### Scenario: Make Reservation

<b>Use Case:</b>	Make Reservation
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest confirmed their selected room and wrote their information.
<b>Preconditions:</b>	The guest selected a room and entered their valid information.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest confirms the room they have selected with their details.</li> <li>2. The system reserves the room for the guests and confirms the reservation.</li> </ol>
<b>Exceptions:</b>	

<b>2a.</b>	The room becomes unavailable during the reservation process.
	<ol style="list-style-type: none"> <li>1. System notified guest</li> <li>2. The process is halted.</li> </ol>

### Scenario: Process Payment

<b>Use Case:</b>	Process Payment
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest finalizing their booking process with payment for the reservation.
<b>Preconditions:</b>	The guest successfully made the reservation after room selection and personal information.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest selects their preferred payment method (PayPal, credit card, etc).</li> <li>2. The guest enters their payment details (card number, full name, CVS).</li> <li>3. The system would process the payment</li> <li>4. The system would send an authentication code to the guest to verify.</li> <li>5. The guest enters the authenticated code that was sent by the bank.</li> <li>6. The system confirms the payment was successful and sends a receipt to the guest</li> </ol>
<b>Exceptions:</b>	
<b>5a.</b>	Payment fails due to insufficient funds or card errors.
	<ol style="list-style-type: none"> <li>3. The system notifies the guest.</li> <li>4. The system asks the user to retry or use a different method.</li> </ol>

### Scenario: Reservation Confirmation

<b>Use Case:</b>	Reservation Confirmation
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest completes their payment successfully.
<b>Preconditions:</b>	The payment has been processed.

<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The system generates a reservation confirmation with the booking details for the guest.</li> <li>2. The system would also generate a booking reference with the guest's details.</li> <li>3. The system sends the invoice and the booking reference to the guest (via email).</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	Confirmation has not been sent due to technical issues.
	<ol style="list-style-type: none"> <li>1. The guest would contact support to inform the system's customer support that the email was not sent.</li> <li>2. The system logs the issue.</li> </ol>

#### Scenario: Access Account

<b>Use Case:</b>	Access Account
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to view or manage their reservation
<b>Preconditions:</b>	The guest has a registered account and is logged in
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest clicks on the account login page.</li> <li>2. The guest enters their valid login details (email address/username and password).</li> <li>3. The system authenticates this and successfully logs into the guests account.</li> <li>4. The system displays the guest's account details.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	Incorrect login
	<ol style="list-style-type: none"> <li>1. The system prompts the guest to retry or reset their password.</li> </ol>

#### Scenario: Authenticate User

<b>Use Case:</b>	Authenticate User
<b>Actors:</b>	Guests and Hotel Reservation System

<b>Trigger:</b>	The guest tries to log into the reservation system
<b>Preconditions:</b>	The guest has to have an account.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest enters their username/email and their password.</li> <li>2. The system validates this.</li> <li>3. The system logs the guests into their accounts,.</li> </ol>
<b>Exceptions:</b>	
<b>2a.</b>	Incorrect login
	<ol style="list-style-type: none"> <li>2. The system prompts the guest to retry or reset their password.</li> </ol>

#### Scenario: Retrieve Reservation Details

<b>Use Case:</b>	Retrieve Reservation Details
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to view the details of their current reservation.
<b>Preconditions:</b>	The guest is logged into their account and has a reservation
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest selects the menu to view their reservation.</li> <li>2. The system retrieves that reservation with its details from the database.</li> <li>3. The system displays the reservation for the guest to view.</li> </ol>
<b>Exceptions:</b>	
<b>2a.</b>	The system cannot retrieve the reservation details due to a system error.
	<ol style="list-style-type: none"> <li>1. The system notifies the guest of the issue and asks them to refresh or try again later.</li> <li>2. If the problem persists, the system would suggest the guest to contact costumer support.</li> </ol>

#### Scenario: Retrieve Print Reservation

<b>Use Case:</b>	Print Reservation
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to print/save the invoice of their reservation to their

	desktop or as a hardcover.
<b>Preconditions:</b>	The guest has to have access to their reservation
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>4. The guest selects the menu to view their reservation.</li> <li>5. The system retrieves that reservation with its details from the database.</li> <li>6. The system displays the reservation for the guest to view.</li> </ol>
<b>Exceptions:</b>	
<b>2a.</b>	The system cannot retrieve the reservation details due to a system error.
	<ol style="list-style-type: none"> <li>3. The system notifies the guest of the issue and asks them to refresh or try again later.</li> <li>4. If the problem persists, the system would suggest the guest to contact customer support.</li> </ol>

#### Scenario: Update Booking Details

<b>Use Case:</b>	Update Booking Details
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to update/modify their reservation
<b>Preconditions:</b>	The guest has to have access to their account and their reservation
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest clicks on the option on their reservation to update their booking.</li> <li>2. The system displays the booking details and a bar where the guests can edit (e.g., room type, date, etc).</li> <li>3. The guests edit their booking where it is allowed.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	The update conflicts with the availability of the room or the hotel's policy.
	<ol style="list-style-type: none"> <li>1. The system notifies the guest of that and would suggest a different timeline or contact the hotel.</li> </ol>

#### Scenario: Modify Reservation

<b>Use Case:</b>	Modify Reservation
------------------	--------------------



<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to make changes to their booking
<b>Preconditions:</b>	The guest has updated their booking details.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest confirms the updated booking.</li> <li>2. The system processes the modification.</li> <li>3. The system updates the reservation..</li> <li>4. The system confirms the updated reservation.</li> </ol>
<b>Exceptions:</b>	
<b>4a.</b>	The update conflicts with the availability of the room or the hotel's policy.
	<ol style="list-style-type: none"> <li>2. The system notifies the guest of that and would suggest a different timeline or contact the hotel.</li> </ol>

#### Scenario: Process Additional Payment

<b>Use Case:</b>	Process Additional Payment
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guests need to make an additional payment for the reservation modification.
<b>Preconditions:</b>	The guest successfully modified their reservation.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The system would calculate the additional charge based on the modification.</li> <li>2. The guest selects their payment method.</li> <li>3. The guest enters their payment details (name, card type, CVS).</li> <li>4. The systems process the payment.</li> <li>5. The system confirms the payment is successful.</li> </ol>
<b>Exceptions:</b>	
<b>5a.</b>	The payment is insufficient due to fund issues or gateway issues.
	<ol style="list-style-type: none"> <li>3. The system notifies the guest of that and suggests trying or using a different method.</li> </ol>

#### Scenario: Request Cancellation

<b>Use Case:</b>	Request Cancellation
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest wants to cancel their reservation.
<b>Preconditions:</b>	The guest is logged in and has a current reservation.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest selects the option to cancel the reservation.</li> <li>2. The system asks the guest to enter their booking reference</li> <li>3. The system retrieves the reservation for cancellation.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	The reservation does not meet the hotel's cancellation policy.
	<ol style="list-style-type: none"> <li>1. The system notifies the guest that this does not meet the hotel's policy.</li> </ol>

#### Scenario: Add Booking Reference

<b>Use Case:</b>	Add Booking Reference
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest enters the booking references to cancel their reservation.
<b>Preconditions:</b>	The guest requested the cancellation of the reservation.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The guest enters their booking reference to the system.</li> <li>2. The system validates the booking reference</li> </ol>
<b>Exceptions:</b>	
<b>2a.</b>	The reservation booking is invalid, or the system cannot find it.
	<ol style="list-style-type: none"> <li>1. The system would ask the guest to retry/ provide a valid booking reference.</li> </ol>

#### Scenario: Confirm Cancellation

<b>Use Case:</b>	Confirm Cancellation
<b>Actors:</b>	Guests and Hotel Reservation System

<b>Trigger:</b>	The guests accept that they want to proceed with the cancellation.
<b>Preconditions:</b>	The booking reference is entered correctly and is valid.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The system asks again if the guest wants to proceed with their cancelation request.</li> <li>2. The guest confirms that they want to cancel the reservation.</li> <li>3. The system processes the cancellation.</li> </ol>
<b>Exceptions:</b>	
<b>1a.</b>	The guest changes their mind and does not want to cancel their reservation.
	<ol style="list-style-type: none"> <li>1. The system exits the cancellation prompt and does not process the cancellation.</li> </ol>

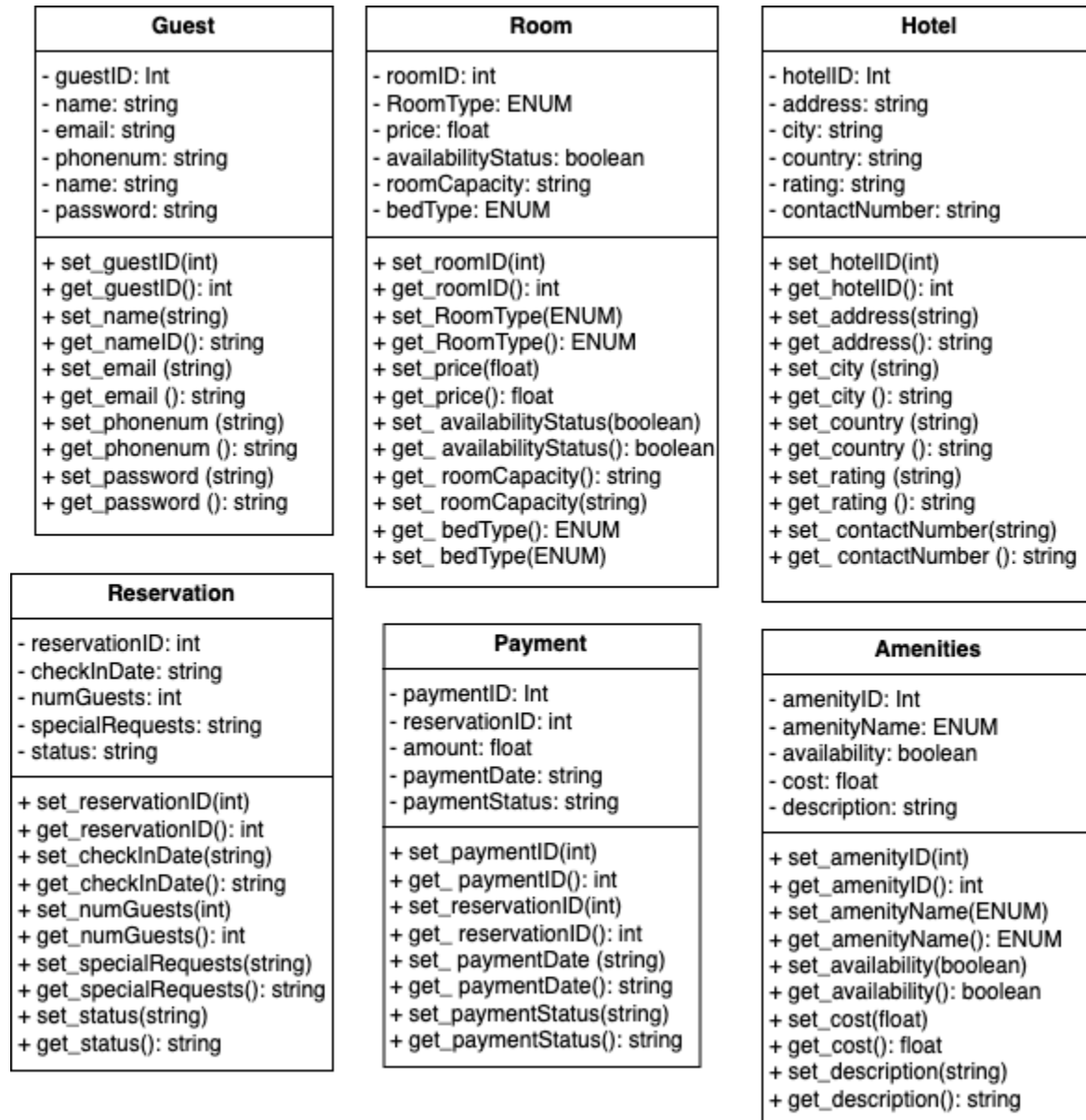
#### Scenario: Cancel Reservation

<b>Use Case:</b>	Cancel Reservation
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guests accept the reservation cancellation
<b>Preconditions:</b>	The guest confirmed their intent to cancel the request, and the system validated it.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The system cancels the reservation</li> <li>2. The system goes back to the database and updates that the canceled room is now available for others to book.</li> <li>3. The system sends a cancellation confirmation to the guest via email.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	The system cannot cancel the reservation due to technical issues within the hotel's system.
	<ol style="list-style-type: none"> <li>1. The system informs the guest that there was an error in the system and asks them to try again later or contact customer support.</li> </ol>

#### Scenario: Process Refund

<b>Use Case:</b>	Process Refund
<b>Actors:</b>	Guests and Hotel Reservation System
<b>Trigger:</b>	The guest canceled their reservation and is allowed/eligible to get their refund.
<b>Preconditions:</b>	The reservation was canceled successfully, and the guest is in the process to get their refund.
<b>Main Scenarios:</b>	<ol style="list-style-type: none"> <li>1. The system calculates the refund amount based on the hotel's policy.</li> <li>2. The system processes the refund for the guest via their payment method.</li> <li>3. The system sends an email to the guest with their refund confirmation.</li> </ol>
<b>Exceptions:</b>	
<b>3a.</b>	The refund process fails due to technical issues or payment gateway errors.
	<ol style="list-style-type: none"> <li>2. The system informs the guest that there was an error in the system and asks them to try again later or contact customer support.</li> </ol>

## UML Class Diagram:



## UML Class Description:

### 1. Guest Class:

#### a. Attributes:

- i. guestID: int
- ii. name: string
- iii. email: string
- iv. phonenum: string
- v. password: string

#### b. Methods:

- i. + set\_guestID(int)
- ii. + get\_guestID(): int
- iii. + set\_name(string)
- iv. + get\_name(): string
- v. + set\_email(string)
- vi. + get\_email(): string
- vii. + set\_phonenum(string)
- viii. + get\_phonenum(): string
- ix. + set\_password(string)
- x. + get\_password(): string

### 2. Room Class:

#### a. Attributes:

- i. roomID: int
- ii. RoomType: string
- iii. price: float
- iv. availabilityStatus: boolean

- v. roomCapacity: int
- vi. bedType: ENUM

**b. Methods:**

- i. + set\_roomID(int)
- ii. + get\_roomID(): int
- iii. + set\_RoomType(ENUM)
- iv. + get\_RoomType(): ENUM
- v. + set\_price(float)
- vi. + get\_price(): float
- vii. + set\_availabilityStatus(boolean)
- viii. + get\_availabilityStatus(): boolean
- ix. + set\_roomCapacity(boolean)
- x. + get\_roomCapacity(): boolean
- xi. + set\_bedType(ENUM)
- xii. + get\_bedType(): ENUM

**3. Hotel Class:**

**a. Methods:**

- i. hotelID: int
- ii. address: string
- iii. city: string
- iv. country: string
- v. rating: string
- vi. contactNumber: string

**b. Methods:**

- i. + set\_hotelID(int)
- ii. + get\_hotelID(): int

- iii. + set\_address(string)
- iv. + get\_address(): string
- v. + set\_city(string)
- vi. + get\_city(): string
- vii. + set\_country(string)
- viii. + get\_country(): string
- ix. + set\_rating(string)
- x. + get\_rating(): string
- xi. + set\_contactNumber(string)
- xii. + get\_contactNumber(): string

#### **4. Reservation Class:**

##### **a. Attributes:**

- i. reservationID: int
- ii. checkInDate: string
- iii. numGuests: int
- iv. specialRequests: string
- v. status: string

##### **b. Methods:**

- i. + set\_reservationID(int)
- ii. + get\_reservationID(): int
- iii. + set\_checkInDate(string)
- iv. + get\_checkInDate(): string
- v. + set\_numGuests(int)
- vi. + get\_numGuests(): int
- vii. + set\_specialRequests(string)



- viii. + get\_specialRequests(): string
- ix. + set\_status(string)
- x. + get\_status(): string

## **5. Payment Class:**

### **a. Attributes:**

- i. paymentID: int
- ii. reservationID: int
- iii. amount: float
- iv. paymentDate: string
- v. paymentStatus: string

### **b. Methods:**

- i. + set\_paymentID(int)
- ii. + get\_paymentID(): int
- iii. + set\_reservationID(int)
- iv. + get\_reservationID(): int
- v. + set\_paymentDate(string)
- vi. + get\_paymentDate(): string
- vii. + set\_paymentStatus(string)
- viii. + get\_paymentStatus(): string

## **6. Amenities Class:**

### **a. Attributes:**

- i. amenityID: int
- ii. amenityName: ENUM
- iii. availability: boolean
- iv. cost: float
- v. description: string

**b. Methods:**

- i. + set\_amenityID(int)
- ii. + get\_amenityID(): int
- iii. + set\_amenityName(ENUM)
- iv. + get\_amenityName(): ENUM
- v. + set\_availability(boolean)
- vi. + get\_availability(): boolean
- vii. + set\_cost(float)
- viii. + get\_cost(): float
- ix. + set\_description(string)
- x. + get\_description(): string

## Code Cell:

```
from enum import Enum

class RoomType(Enum): #boolean statement for the room type
    """class for room types for guest's reservation"""
    JUNIOR_SUITE = "Junior Suite"
    DELUXE_SUITE = "Deluxe Suite"
    SUPERIOR_ROOM = "Superior Room"
    DELUXE_GUESTROOM = "Deluxe Guestroom"
    GUEST_ROOM = "Guest Room"

class BedType(Enum): #boolean statement for the bed type
    """class for bed types for guest's reservation"""
    KING = "King"
    TWIN = "Twin"
    QUEEN = "Queen"

class AmenityType(Enum): #boolean statement for the amenity
    """class for amenity for guest's reservation"""
    FREE_INTERNET = "Free Internet"
    POOL = "Pool"
    GYM = "Gym"
    PARKING = "Parking"
    SPA_SERVICE = "Full Service Spa"
    BREAKFAST = "Breakfast"

# Guest Class
class Guest:
```

```
"""class for Guests in Online Hotel Reservation"""
```

```
# constructor to define the attributes
```

```
def __init__(self, guestID, name, email, phonenumber, password):
```

```
    self.__guestID = guestID
```

```
    self.__name = name
```

```
    self.__email = email
```

```
    self.__phonenumber = phonenumber
```

```
    self.__password = password
```

```
# Setter and getter methods for each attribute
```

```
def get_guestID(self):
```

```
    return self.__guestID
```

```
def set_guestID(self, guestID):
```

```
    self.__guestID = guestID
```

```
def get_name(self):
```

```
    return self.__name
```

```
def set_name(self, name):
```

```
    self.__name = name
```

```
def get_email(self):
```

```
    return self.__email
```

```
def set_email(self, email):
```

```
    self.__email = email
```

```
def get_phonenum(self):  
    return self.__phonenum  
  
def set_phonenum(self, phonenum):  
    self.__phonenum = phonenum  
  
def get_password(self):  
    return self.__password  
  
def set_password(self, password):  
    self.__password = password  
  
# __str__ method for the object  
def __str__(self):  
    return f"Guest[ID: {self.__guestID}, Name: {self.__name}, Email:  
{self.__email}, Phone: {self.__phonenum}]"  
  
# Function headers with pass statements  
def loginVerification(self):  
    # Verify guest's login information  
    pass  
  
def reservationDisplay(self):  
    # Displays the guest's reservation booking details  
    pass  
  
def detailUpdated(self):  
    # Updates guest's information details  
    pass
```

```
# Room class

class Room:

    """class for Rooms in Online Hotel Reservation"""

    # constructor to define the attributes

    def __init__(self, roomID, roomType, price, availabilityStatus,
roomCapacity, bedType):

        self.__roomID = roomID

        self.__roomType = roomType

        self.__price = price

        self.__availabilityStatus = availabilityStatus

        self.__roomCapacity = roomCapacity

        self.__bedType = bedType

        self.__description = ""

    # Getter and Setter methods

    def set_roomID(self, roomID):

        self.__roomID = roomID

    def get_roomID(self):

        return self.__roomID

    def set_roomType(self, roomType):

        self.__roomType = roomType

    def get_roomType(self):

        return self.__roomType
```

```
def set_price(self, price):
    self.__price = price

def get_price(self):
    return self.__price

def set_availabilityStatus(self, availabilityStatus):
    self.__availabilityStatus = availabilityStatus

def get_availabilityStatus(self):
    return self.__availabilityStatus

def set_roomCapacity(self, roomCapacity):
    self.__roomCapacity = roomCapacity

def get_roomCapacity(self):
    return self.__roomCapacity

def set_bedType(self, bedType):
    self.__bedType = bedType

def get_bedType(self):
    return self.__bedType

# __str__ method for object
def __str__(self):
    return f"Room[ID: {self.__roomID}, Type: {self.__roomType.value}, Price: {self.__price}, Available: {self.__availabilityStatus}]"
```

```
# Methods for specific room types

def set_junior_suite(self):

    self.__roomType(RoomType.JUNIOR_SUITE)

    self.__price = 800.0

    self.__roomCapacity = 3

    self.__bedType((BedType.QUEEN))

    self.__description = "A large and spacious junior suite with a separate
living area."

def set_deluxe_suite(self):

    self.set_roomType(RoomType.DELUXE_SUITE)

    self.__price = 1200.0

    self.__roomCapacity = 4

    self.set_bedType(BedType.KING)

    self.__description = "Enjoy our luxurious deluxe suite with premium
amenities."

def set_superior_room(self):

    self.set_roomType(RoomType.SUPERIOR_ROOM)

    self.__price = 700.0

    self.__roomCapacity = 2

    self.set_bedType(BedType.QUEEN)

    self.__description = "A superior room with great comfort."

def set_deluxe_guestroom(self):

    self.set_roomType(RoomType.DELUXE_GUESTROOM)

    self.__price = 750.0

    self.__roomCapacity = 2

    self.set_bedType(BedType.KING)
```



```

        self.__description = "A deluxe guestroom with extra space and premium
bedding."

    def set_guest_room(self):
        self.set_roomType(RoomType.GUEST_ROOM)
        self.__price = 500.0
        self.__roomCapacity = 2
        self.set_bedType(BedType.TWIN)
        self.__description = "A comfortable guest room suitable."

# Methods for specific bed types
    def set_king_bed(self):
        self.set_bedType(BedType.KING)

    def set_twin_beds(self):
        self.set_bedType(BedType.TWIN)

    def set_queen_bed(self):
        self.set_bedType(BedType.QUEEN)

# Reservation class
class Reservation:
    """class for Reservation in Online Hotel Reservation"""

    # constructor to define the attributes
    def __init__(self, reservationID, checkInDate, numGuests, specialRequests,
status):
        self.__reservationID = reservationID

```

```
self.__checkInDate = checkInDate

self.__numGuests = numGuests

self.__specialRequests = specialRequests

self.__status = status


# Getter and Setter methods

def set_reservationID(self, reservationID):

    self.__reservationID = reservationID


def get_reservationID(self):

    return self.__reservationID


def set_checkInDate(self, checkInDate):

    self.__checkInDate = checkInDate


def get_checkInDate(self):

    return self.__checkInDate


def set_numGuests(self, numGuests):

    self.__numGuests = numGuests


def get_numGuests(self):

    return self.__numGuests


def set_specialRequests(self, specialRequests):

    self.__specialRequests = specialRequests


def get_specialRequests(self):

    return self.__specialRequests
```

```

def set_status(self, status):

    self.__status = status


def get_status(self):

    return self.__status


# __str__ method for object
def __str__(self):

    return f"Reservation[ID: {self.__reservationID}, Guests: {self.__numGuests}, Status: {self.__status}]"


# Function headers with pass statements
def reserveconfirm(self): # The guest's reservation confirmation

    pass


def Reservenew(self): # Create a new reservation for the guest

    pass


def Reservemodify(self): # Modifying the existing reservation made by the
guest

    pass


def cancelreserve(self): # Cancels a reservation

    pass


# Payment class
class Payment:

```

```
"""class for Payment in Online Hotel Reservation"""

# constructor to define the attributes

    def __init__(self, paymentID, reservationID, amount, paymentDate,
paymentStatus):

    self.__paymentID = paymentID

    self.__reservationID = reservationID

    self.__amount = amount

    self.__paymentDate = paymentDate

    self.__paymentStatus = paymentStatus

# Getter and Setter methods

def set_paymentID(self, paymentID):

    self.__paymentID = paymentID

def get_paymentID(self):

    return self.__paymentID

def set_reservationID(self, reservationID):

    self.__reservationID = reservationID

def get_reservationID(self):

    return self.__reservationID

def set_amount(self, amount):

    self.__amount = amount

def get_amount(self):

    return self.__amount
```

```

def set_paymentDate(self, paymentDate):
    self.__paymentDate = paymentDate

def get_paymentDate(self):
    return self.__paymentDate

def set_paymentStatus(self, paymentStatus):
    self.__paymentStatus = paymentStatus

def get_paymentStatus(self):
    return self.__paymentStatus

# Function headers with pass statements

def paymentprocess(self):    # Processes the guest's payment for their
reservation
    pass

def refund(self):    # Refund the guest's payment for their booking
cancellation
    pass

# __str__ method for the object
def __str__(self):
    return f"Payment[ID: {self.__paymentID}, Amount: {self.__amount},
Status: {self.__paymentStatus}]"

# Amenities Class
class Amenities:

```

```
"""Class for Amenities in Online Hotel Reservation"""

# constructor to define the attributes

def __init__(self, amenityID, amenityName, availability, cost, description):
    self.__amenityID = amenityID
    self.__amenityName = amenityName
    self.__availability = availability
    self.__cost = cost
    self.__description = description

# Getter and Setter methods

def set_amenityID(self, amenityID):
    self.__amenityID = amenityID

def get_amenityID(self):
    return self.__amenityID

def set_amenityName(self, amenityName):
    self.__amenityName = amenityName

def get_amenityName(self):
    return self.__amenityName

def set_availability(self, availability):
    self.__availability = availability

def get_availability(self):
    return self.__availability
```

```

def set_cost(self, cost):
    self.__cost = cost

def get_cost(self):
    return self.__cost

def set_description(self, description):
    self.__description = description

def get_description(self):
    return self.__description

# __str__ method for easy object representation
def __str__(self):
    return f"Amenities[ID: {self.__amenityID}, Name: {self.__amenityName},
Cost: {self.__cost}]"

#methods for specific amenity types
def free_internet(self): #free wifi
    self.set_amenityName(AmenityType.FREE_INTERNET)
    self.__availability = True
    self.__cost = 0.0 #free
    self.__description = "Complimentary internet during your stay at our
hotel."

#Hotel having a pool access for guests
def pool(self):
    self.set_amenityName(AmenityType.POOL)
    self.__availability = True

```

```

        self.__cost = 0.0 #free for guests

        self.__description = "Outdoor swimming pool available for guests."

#gym hotel
def gym(self):

    self.set_amenityName(AmenityType.GYM)

    self.__availability = True

    self.__cost = 0.0 # free for guests

    self.__description = "24/7 access to the hotel gym for guests."

#hotel parking
def parking(self):

    self.set_amenityName(AmenityType.PARKING)

    self.__availability = True

    self.__cost = 50.0 #cost per day

    self.__description = "Parking for guests."

#spa service
def spa_service(self):

    self.set_amenityName(AmenityType.SPA_SERVICE.value)

    self.__availability = True

    self.__cost = 100.0 # per service

    self.__description = "Enjoy our spa services which includes massages,
facials, and more."

def breakfast(self, free = 50.0):

    self.set_amenityName(AmenityType.BREAKFAST)

    self.__availability = True

    self.__cost = free # Fee can be adjusted

```



```
        self.__description = f"Complimentary hot breakfast available for an  
additional fee starting from {fee} EUR."  
  
# Hotel Class  
class Hotel:  
    """Class for Hotel in Online Hotel Reservation"""  
  
    # constructor to define the attributes  
    def __init__(self, hotelID, address, city, country, rating, contactNumber):  
        self.__hotelID = hotelID  
        self.__address = address  
        self.__city = city  
        self.__country = country  
        self.__rating = rating  
        self.__contactNumber = contactNumber  
  
    # Getter and Setter methods  
    def set_hotelID(self, hotelID):  
        self.__hotelID = hotelID  
  
    def get_hotelID(self):  
        return self.__hotelID  
  
    def set_address(self, address):  
        self.__address = address  
  
    def get_address(self):  
        return self.__address
```

```
def set_city(self, city):
    self.__city = city

def get_city(self):
    return self.__city

def set_country(self, country):
    self.__country = country

def get_country(self):
    return self.__country

def set_rating(self, rating):
    self.__rating = rating

def get_rating(self):
    return self.__rating

def set_contactNumber(self, contactNumber):
    self.__contactNumber = contactNumber

def get_contactNumber(self):
    return self.__contactNumber

# __str__ method for easy object representation
def __str__(self):
    return f"Hotel[ID: {self.__hotelID}, Address: {self.__address}, City: {self.__city}, Country: {self.__country}, Rating: {self.__rating}]"
```

```
# Object for Guest

guest1 = Guest(guestID=101, name="Ahmed Abdulla", email="almoallaa@gmail.com",
phonenum="0507895455", password="Ahmed1234")

print(guest1)


# Object for Room

room1 = Room(roomID=501, roomType="", price=0.0, availabilityStatus=True,
roomCapacity=0, bedType="")

room1.set_deluxe_suite() # Setting room as a Deluxe Suite
room1.set_king_bed()     # Setting the bed type to King

print(room1)


#object for reservation

reservation1 = Reservation(reservationID=2133, checkInDate= "30-09-2024",
numGuests=3, specialRequests="Replace feather pillows and late check in",
status= "confirmed")

print(reservation1)


#object for payment

payment1 = Payment(paymentID=5400, reservationID=2133, amount=1200.0,
paymentDate="29-09-2024", paymentStatus="Completed")

print(payment1)


# Object for Amenities

amenity1 = Amenities(amenityID=5001, amenityName="", availability=False,
cost=0.0, description="")

amenity1.spa_service() # Setting the amenity as Spa Service

print(amenity1)
```

```
# Object for Hotel

hotel1 = Hotel(hotelID=1, address="Bayerstraße 41", city="Münich",
country="Germany", rating=4, contactNumber="+498924220")

print(hotel1)
```

**Link to GitHub:**

<https://github.com/hindalmualla/Hotel-Reservation-System.git>

**Summery:**

Throughout this assignment I have implement object-oriented principles beginning with creating UML use-case diagrams and descriptions, and then creating UML Class Diagram and its descriptions; encompassing all that, created a code for the hotel reservation scenario.

I have defined several classes (Guest, Room, Hotel, Reservation, Payment, and Amenities) to represent different entities in an online hotel reservation system, and each having their own attributes (like guestID, room type, paymentID, etc) and behaviors with the methods. I have implemented the concept of encapsulation by using the setter and getter methods for each attributes to control the data. I have also implemented \_\_str\_\_ method in each class to display each of the object's output. I have alos used ENUM for specific sets fo values like (room type, bed type, AmenityType) for making it a bit more structured in a way.

Throughout the code, I hope it is well-documented as I have included docstrings and comments, explaining the purpose of each class and functions, for ease of understanding of the code. I have also structured the code that is readable and understandable.

I have stumbled upon some errors on python, especially in the output where instead of displaying the room type "Deluxe Suite" it printed RoomType.DELUXE\_SUITE. Therefore, to fix this bug, I had only to add .value to print "Delux Suite". In the end, I hope I designed a code that is free of errors and realistically emulates a real-world reservation system— in congruent to OOP practice.