**Hotel Reservation System**

Sheikha H. Almoalla

College of Interdisciplinary Studies, Zayed University

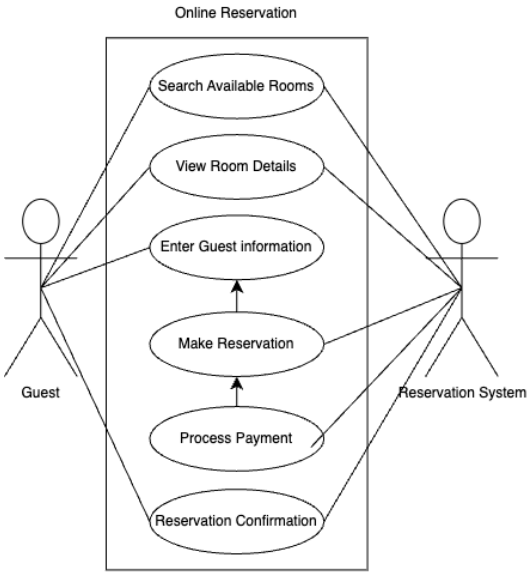
22407 Ethical System Moral Dilemmas

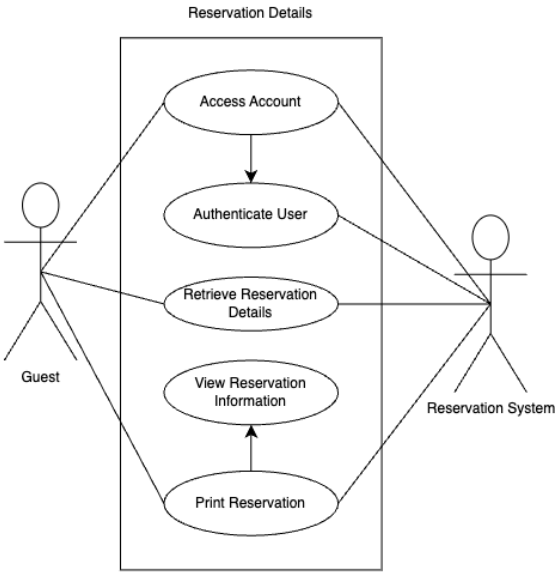
Dr. Amir Kaviani

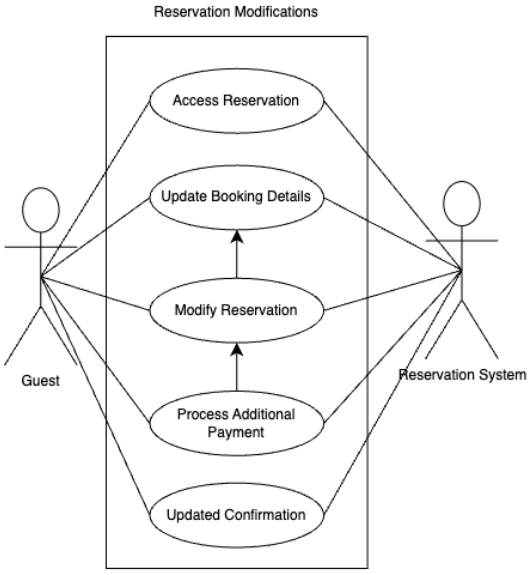
September 20, 2024

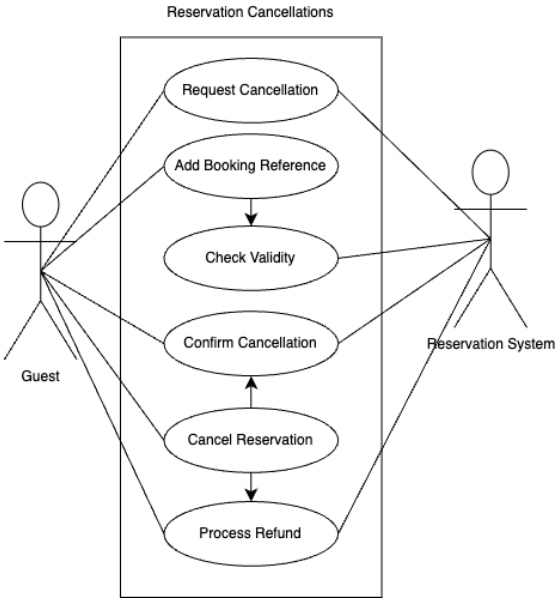
# Hotel Reservation System

## Use-Case Diagrams:



**Use-Case Descriptions:**

**Scenario: Room Availability**

| | |
|---|---|
| **Use Case:** | Search Available Rooms |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | Guests want to book a room, so they start by searching for available rooms from the hotel's system |
| **Preconditions:** | The rooms are available on the hotel's reservation system |
| **Main Scenarios:** | 1. Guests are searching on the online reservation system of the hotel<br>2. Guests would enter their criteria or preferences through the filter bar (e.g., dates, room type, number of guests).<br>3. The hotel's system would process the query.<br>4. The hotel's system would retrieve and display the available rooms based on the guest's preferences. |
| **Exceptions:** | |
| **4a.** | No rooms available that match the guest's preferences |
| | 1. The system informs the guest.<br>2. The system suggests modifying the search criteria. |

**Scenario: View Room Details**

| | |
|---|---|
| **Use Case:** | View Room Details |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | Guests select a specific room they are interested in from the list available and view the rooms' detailed information |
| **Preconditions:** | The guest has performed a room search, and room information is displayed in the hotel's system. |
| **Main Scenarios:** | 1. The guests select their preferred room and view the details.<br>2. The system retrieves the information of the room (i.,e price, photos, bed size, and other amenities)<br>3. The system would display that to the guests. |
| **Exceptions:** | |

| | |
|---|---|
| **3a.** | The selected room is no longer available. |
| | 1. The system informs the guest of the misfortune.<br>2. The system suggests going back and searching for the other available rooms. |

**Scenario: Enter Guest Information**

| | |
|---|---|
| **Use Case:** | Enter Guest Information |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | Guests found a suitable room and proceeded to check out. |
| **Preconditions:** | The guests have to select rooms and proceed with check-out |
| **Main Scenarios:** | 1. The guests would enter their personal information (special requests, full name, email address, and contact information).<br>2. The guests would review the information.<br>3. The system would validate the information by sending a confirmation email.<br>4. The guest would authenticate it via email. |
| **Exceptions:** | |
| **3a.** | Information is not valid (e.g., incorrect email address) or incomplete (e.g., missing phone number) |
| | 1. The system would prompt the guest to correct the information. |

**Scenario: Make Reservation**

| | |
|---|---|
| **Use Case:** | Make Reservation |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest confirmed their selected room and wrote their information. |
| **Preconditions:** | The guest selected a room and entered their valid information. |
| **Main Scenarios:** | 1. The guest confirms the room they have selected with their details.<br>2. The system reserves the room for the guests and confirms the reservation. |
| **Exceptions:** | |

| | |
|---|---|
| **2a.** | The room becomes unavailable during the reservation process. |
| | 1. System notified guest<br>2. The process is halted. |

**Scenario: Process Payment**

| | |
|---|---|
| **Use Case:** | Process Payment |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest finalizing their booking process with payment for the reservation. |
| **Preconditions:** | The guest successfully made the reservation after room selection and personal information. |
| **Main Scenarios:** | 1. The guest selects their preferred payment method (PayPal, credit card, etc).<br>2. The guest enters their payment details (card number, full name, CVS).<br>3. The system would process the payment<br>4. The system would send an authentication code to the guest to verify.<br>5. The guest enters the authenticated code that was sent by the bank.<br>6. The system confirms the payment was successful and sends a receipt to the guest |
| **Exceptions:** | |
| **5a.** | Payment fails due to insufficient funds or card errors. |
| | 3. The system notifies the guest.<br>4. The system asks the user to retry or use a different method. |

**Scenario: Reservation Confirmation**

| | |
|---|---|
| **Use Case:** | Reservation Confirmation |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest completes their payment successfully. |
| **Preconditions:** | The payment has been processed. |

| | |
|---|---|
| **Main Scenarios:** | 1. The system generates a reservation confirmation with the booking details for the guest.<br>2. The system would also generate a booking reference with the guest's details.<br>3. The system sends the invoice and the booking reference to the guest (via email). |
| **Exceptions:** | |
| **3a.** | Confirmation has not been sent due to technical issues. |
| | 1. The guest would contact support to inform the system's customer support that the email was not sent.<br>2. The system logs the issue. |

**Scenario: Access Account**

| | |
|---|---|
| **Use Case:** | Access Account |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to view or manage their reservation |
| **Preconditions:** | The guest has a registered account and is logged in |
| **Main Scenarios:** | 1. The guest clicks on the account login page.<br>2. The guest enters their valid login details (email address/username and password).<br>3. The system authecticates this and successfully logs into the guests account.<br>4. The system displays the guest's account details. |
| **Exceptions:** | |
| **3a.** | Incorrect login |
| | 1. The system prompts the guest to retry or reset their password. |

**Scenario: Authenticate User**

| | |
|---|---|
| **Use Case:** | Authenticate User |
| **Actors:** | Guests and Hotel Reservation System |

| | |
|---|---|
| **Trigger:** | The guest tries to log into the reservation system |
| **Preconditions:** | The guest has to have an account. |
| **Main Scenarios:** | 1. The guest enters their username/email and their password.<br>2. The system validates this.<br>3. The system logs the guests into their accounts,. |
| **Exceptions:** | |
| **2a.** | Incorrect login |
| | 2. The system prompts the guest to retry or reset their password. |

**Scenario: Retrieve Reservation Details**

| | |
|---|---|
| **Use Case:** | Retrieve Reservation Details |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to view the details of their current reservation. |
| **Preconditions:** | The guest is logged into their account and has a reservation |
| **Main Scenarios:** | 1. The guest selects the menu to view their reservation.<br>2. The system retrieves that reservation with its details from the database.<br>3. The system displays the reservation for the guest to view. |
| **Exceptions:** | |
| **2a.** | The system cannot retrieve the reservation details due to a system error. |
| | 1. The system notifies the guest of the issue and asks them to refresh or try again later.<br>2. If the problem persists, the system would suggest the guest to contact costumer support. |

**Scenario: Retrieve Print Reservation**

| | |
|---|---|
| **Use Case:** | Print Reservation |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to print/save the invoice of their reservation to their |

| | |
|---|---|
| | desktop or as a hardcover. |
| **Preconditions:** | The guest has to have access to their reservation |
| **Main Scenarios:** | 4. The guest selects the menu to view their reservation.<br>5. The system retrieves that reservation with its details from the database.<br>6. The system displays the reservation for the guest to view. |
| **Exceptions:** | |
| **2a.** | The system cannot retrieve the reservation details due to a system error. |
| | 3. The system notifies the guest of the issue and asks them to refresh or try again later.<br>4. If the problem persists, the system would suggest the guest to contact costumer support. |

**Scenario: Update Booking Details**

| | |
|---|---|
| **Use Case:** | Update Booking Details |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to update/modify their reservation |
| **Preconditions:** | The guest has to have access to their account and their reservation |
| **Main Scenarios:** | 1. The guest clicks on the option on their reservation to update their booking.<br>2. The system displays the booking details and a bar where the guests can edit (e.g., room type, date, etc).<br>3. The guests edit their booking where it is allowed. |
| **Exceptions:** | |
| **3a.** | The update conflicts with the availability of the room or the hotel's policy. |
| | 1. The system notifies the guest of that and would suggest a different timeline or contact the hotel. |

**Scenario: Modify Reservation**

| | |
|---|---|
| **Use Case:** | Modify Reservation |

| | |
|---|---|
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to make changes to their booking |
| **Preconditions:** | The guest has updated their booking details. |
| **Main Scenarios:** | 1. The guest confirms the updated booking.<br>2. The system processes the modification.<br>3. The system updates the reservation..<br>4. The system confirms the updated reservation. |
| **Exceptions:** | |
| **4a.** | The update conflicts with the availability of the room or the hotel's policy. |
| | 2. The system notifies the guest of that and would suggest a different timeline or contact the hotel. |

**Scenario: Process Additional Payment**

| | |
|---|---|
| **Use Case:** | Process Additional Payment |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guests need to make an additional payment for the reservation modification. |
| **Preconditions:** | The guest successfully modified their reservation. |
| **Main Scenarios:** | 1. The system would calculate the additional charge based on the modification.<br>2. The guest selects their payment method.<br>3. The guest enters their payment details (name, card type, CVS).<br>4. The systems process the payment.<br>5. The system confirms the payment is successful. |
| **Exceptions:** | |
| **5a.** | The payment is insufficient due to fund issues or gateway issues. |
| | 3. The system notifies the guest of that and suggests trying or using a different method. |

**Scenario: Request Cancellation**

| | |
|---|---|
| **Use Case:** | Request Cancellation |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest wants to cancel their reservation. |
| **Preconditions:** | The guest is logged in and has a current reservation. |
| **Main Scenarios:** | 1. The guest selects the option to cancel the reservation.<br>2. The system asks the guest to enter their booking reference<br>3. The system retrieves the reservation for cancellation. |
| **Exceptions:** | |
| **3a.** | The reservation does not meet the hotel's cancellation policy. |
| | 1. The system notifies the guest that this does not meet the hotel's policy. |

**Scenario: Add Booking Reference**

| | |
|---|---|
| **Use Case:** | Add Booking Reference |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest enters the booking references to cancel their reservation. |
| **Preconditions:** | The guest requested the cancellation of the reservation. |
| **Main Scenarios:** | 1. The guest enters their booking reference to the system.<br>2. The system validates the booking reference |
| **Exceptions:** | |
| **2a.** | The reservation booking is invalid, or the system cannot find it. |
| | 1. The system would ask the guest to retry/ provide a valid booking reference. |

**Scenario: Confirm Cancellation**

| | |
|---|---|
| **Use Case:** | Confirm Cancellation |
| **Actors:** | Guests and Hotel Reservation System |

| | |
|---|---|
| **Trigger:** | The guests accept that they want to proceed with the cancellation. |
| **Preconditions:** | The booking reference is entered correctly and is valid. |
| **Main Scenarios:** | 1. The system asks again if the guest wants to proceed with their cancelation request.<br>2. The guest confirms that they want to cancel the reservation.<br>3. The system processes the cancellation. |
| **Exceptions:** | |
| **1a.** | The guest changes their mind and does not want to cancel their reservation. |
| | 1. The system exits the cancellation prompt and does not process the cancellation. |

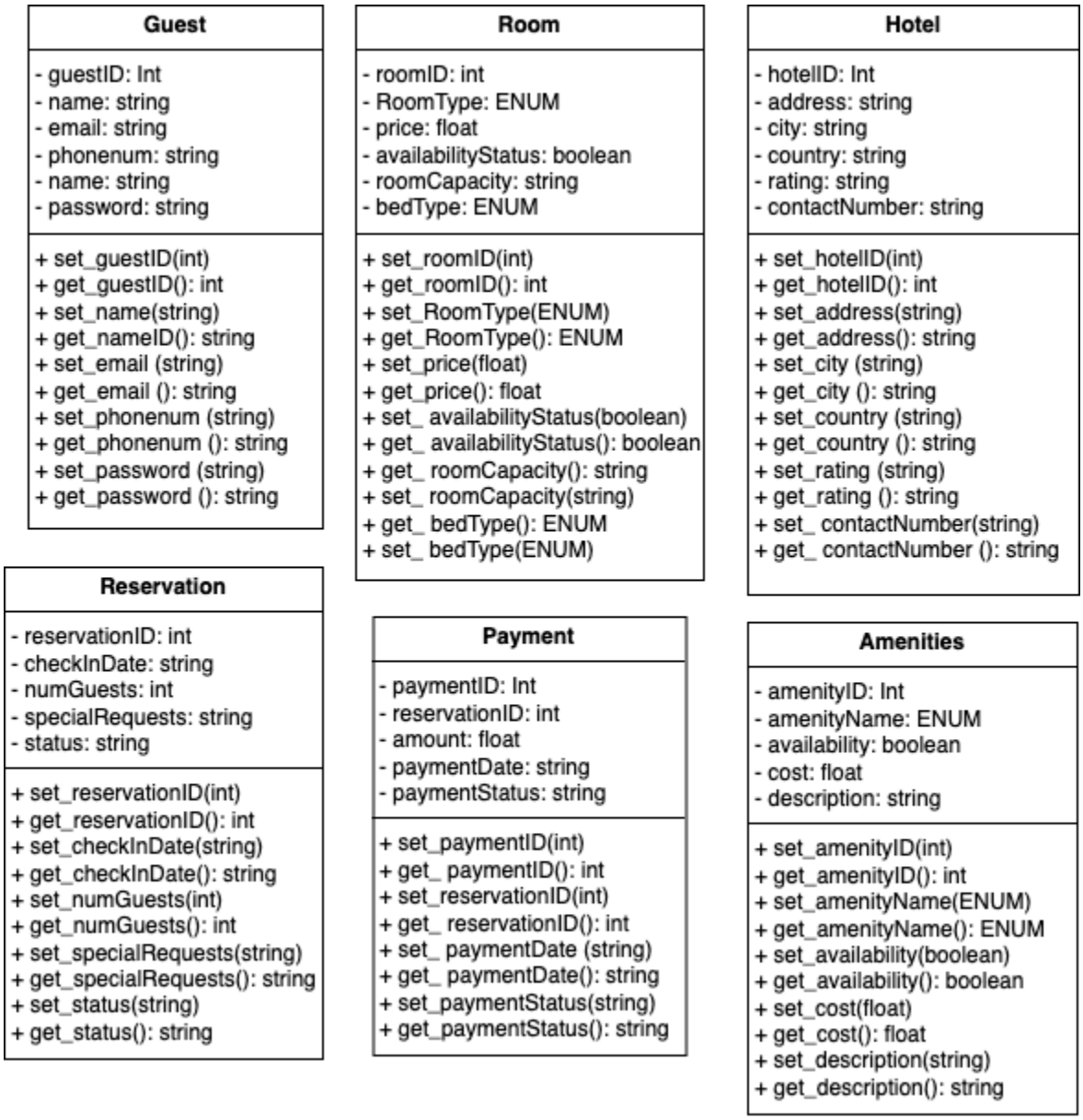
**Scenario: Cancel Reservation**

| | |
|---|---|
| **Use Case:** | Cancel Reservation |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guests accept the reservation cancellation |
| **Preconditions:** | The guest confirmed their intent to cancel the request, and the system validated it. |
| **Main Scenarios:** | 1. The system cancels the reservation<br>2. The system goes back to the database and updates that the canceled room is now available for others to book.<br>3. The system sends a cancellation confirmation to the guest via email. |
| **Exceptions:** | |
| **3a.** | The system cannot cancel the reservation due to technical issues within the hotel's system. |
| | 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. |

**Scenario: Process Refund**

| | |
|---|---|
| **Use Case:** | Process Refund |
| **Actors:** | Guests and Hotel Reservation System |
| **Trigger:** | The guest canceled their reservation and is allowed/eligible to get their refund. |
| **Preconditions:** | The reservation was canceled successfully, and the guest is in the process to get their refund. |
| **Main Scenarios:** | 1. The system calculates the refund amount based on the hotel's policy.<br>2. The system processes the refund for the guest via their payment method.<br>3. The system sends an email to the guest with their refund confirmation. |
| **Exceptions:** | |
| **3a.** | The refund process fails due to technical issues or payment gateway errors. |
| | 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. |

## 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. <u>Hotel Class:</u>**

**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. **<u>Reservation Class:</u>**

   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, phonenum, password):
    self.__guestID = guestID
    self.__name = name
    self.__email = email
    self.__phonenum = phonenum
    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 amentity 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 = fee  # 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)
```

**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-documenetd 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 sum 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.