Hotel Reservation System

Catherine McIlroy, Student No 23173190

Table of Contents

Summary (Task 1)

This report will describe a Hotel Reservation System, and will present various UML diagrams to display the different classes and interactions in the system.

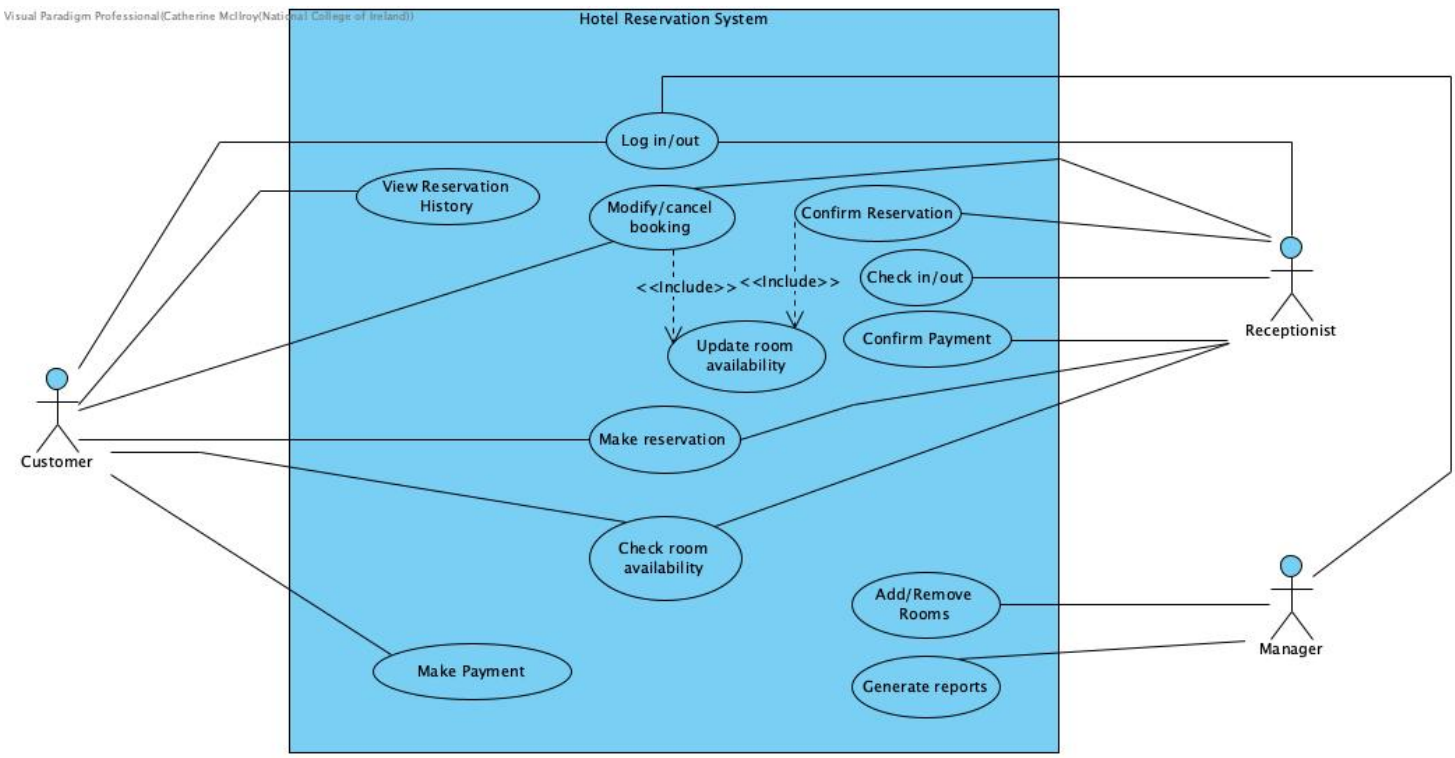
There are several actors involved in this system:

- Customer (Primary actor. The customer requires the system in order to make or modify a reservation.)
  - Log in/out of the reservation system
  - Search for room(s)
  - Book room(s)
  - View, modify or cancel an existing booking
  - Make payment
  - View reservation history

- Receptionist (Secondary actor. The receptionist interacts with the system either on behalf of the customer, or to respond to an action taken by the customer)
  - Log in/out of the reservation system
  - Confirm reservation made by the customer
  - Book room(s), modify an existing booking or cancel a booking on behalf of a customer
  - Check customers in or out of the hotel
  - View booking details
  - Keep a record of payment by the customer

- Manager (Secondary actor. The manager interacts with the system in an administrative capacity only.)
  - Log in/out of the reservation system
  - Can add rooms to or remove rooms from the reservation system
  - Generate reports based on information from the reservation system

Use Case Model (Task 2)



The above use case model depicts the Hotel Reservation System, the different actors involved and their interactions with the system.

All three actors have the ability to log in and out of the system.

The customer and the receptionist share interactions with several of the use cases, including “Modify/Cancel Booking”, “Make Reservation” and “Check Room Availability”; the receptionist would only interact with these on behalf of the customer.

Whenever the “Modify/Cancel Booking” or “Confirm Reservation” use case takes place, the “Update Room Availability” use case is always included. This ensures that the list of available rooms is kept accurate and up to date.

The manager interacts with the administrative system use cases only, i.e. maintaining the room inventory (“Add/Remove Rooms”) and generating reports based on information gathered by the system (“Generate Reports”).

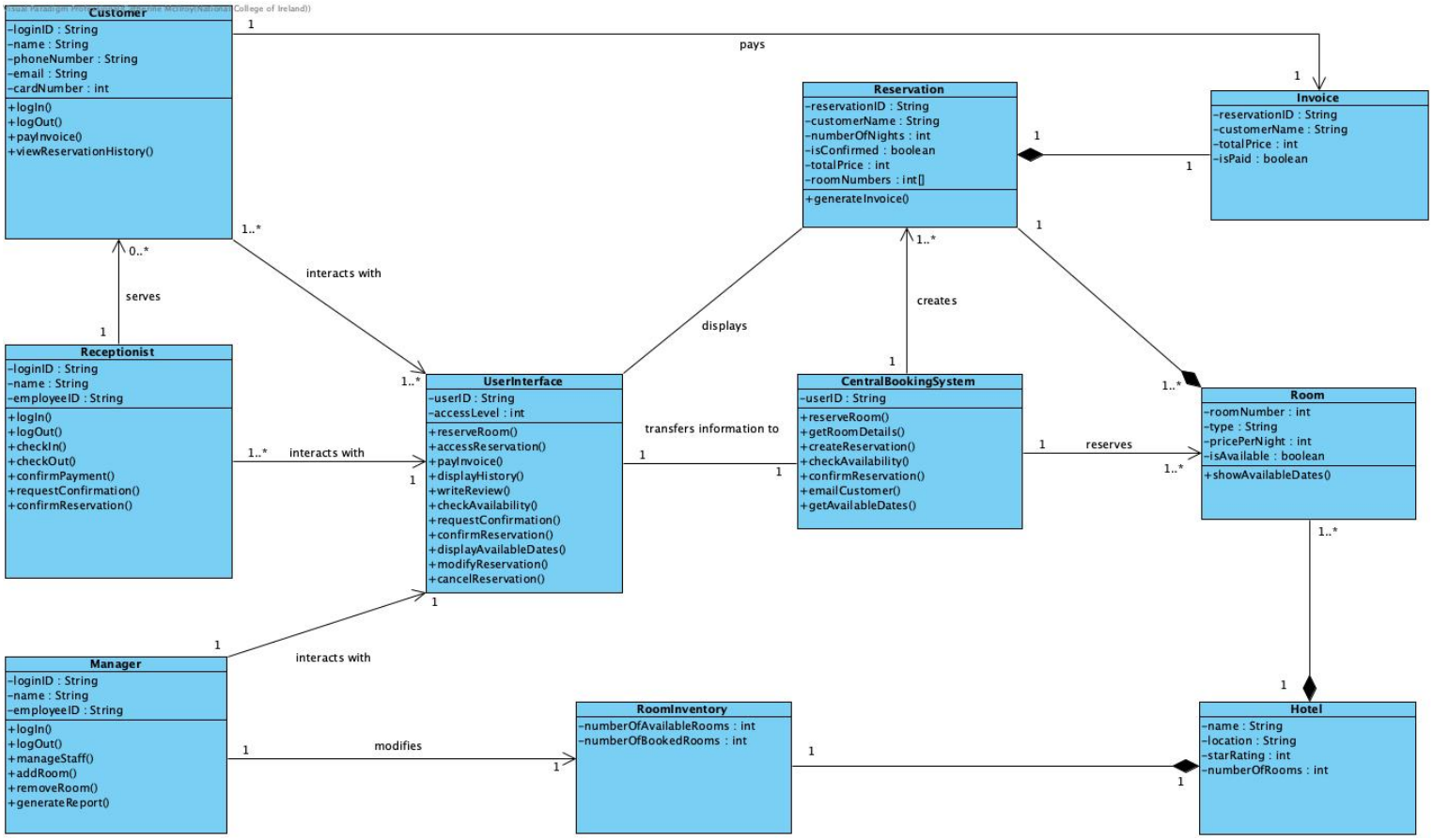
Use Case Description (Task 3)

The “Make Reservation” use case is described in more detail in the following table. I have chosen to examine this particular use case because it is the primary use case of the hotel reservation system.

Use Case Name	Make Reservation
Actors	Customer, Receptionist
Scope	The scope of this use case is for the user to reserve room(s) through the reservation system.
Trigger	Customer accesses the reservation system, or receptionist accesses the reservation system on behalf of a customer
Preconditions	The user must be logged into the system.
Main Flow	1. The user enters the dates of the stay, the number of guests and the number of rooms required 2. The customer is presented with available rooms matching the specified criteria 3. The user selects room(s) to reserve 4. Booking is confirmed 5. Customer receives confirmation correspondence containing reservation details
Alternate Flow	1. 1. The user enters the dates of the stay, the number of guests and the number of rooms required 2. No rooms are available matching the specified criteria 3. The customer is presented with alternative available dates 4. The customer selects an alternative option 5. The process continues from Step 3 of the main flow 2. 1. The user enters the dates of the stay, the number of guests and the number of rooms required 2. No rooms are available matching the specified criteria 3. The customer is presented with alternative available dates 4. The customer decides not to proceed with an alternative option 5. The user exits the reservation system

Exceptional Flow	1. The user enters the dates of the stay, the number of guests and the number of rooms required 2. An error occurs during the search process 3. The system displays an error message to the user 4. The system advises the user on alternate reservation methods 5. The user exits the reservation system
Postconditions	• Main Flow/Alternate Flow 1: - The booking was recorded in the system - Room availability was updated accordingly - Confirmation was sent to the customer • Alternate Flow 2: - No reservation was made - Room availability remains unchanged • Exceptional Flow: - No reservation was made - The user was advised of alternative reservation methods - The system error was logged

Conceptual Class Diagram (Task 4)



The above class diagram shows the various classes which are present in the Hotel Reservation System, their attributes and operations. It depicts the structure of the system, and how these classes interact with one another.

- **CentralBookingSystem class**

  - The main system software, this class performs the operations of the reservation system such as reserving a room, creating a new reservation, and checking room availability.

  - Receives information from the UserInterface class in response to external triggers.

  - One CentralBookingSystem can reserve one or many instances of Room, and can create one or many instances of Reservation class.

- **UserInterface class**
  - Provides an access point for the main system software (CentralBookingSystem).
  - Customer, Receptionist and Manager classes can all interact with the UserInterface class.
  - Transfers information to and receives information from the CentralBookingSystem. Can also interact with Reservation class to display current/past reservation details to the user.
  - Triggers system operations such as “reserveRoom()”, in response to system input events.
  - This class is not involved in any internal system operations. It simply triggers these operations, receives information from the user or the system, and displays information to or requests information from the user.
- **Customer class**
  - One or many instances of customer can interact with one or many instances of UserInterface class. This instance serves as an access point for the system software.
  - One customer pays one instance of Invoice (when a reservation has been confirmed, an instance of Invoice class is created for that reservation).
- **Invoice class**
  - An instance of the Invoice class is generated by the Reservation class when a new reservation is confirmed, i.e. : Reservation attribute isConfirmed = true.
  - Shows a composition relationship with Reservation class; if there is no reservation made then there is no associated invoice.
  - Has a one to one relationship with Reservation class. Each confirmed reservation is associated with one invoice.

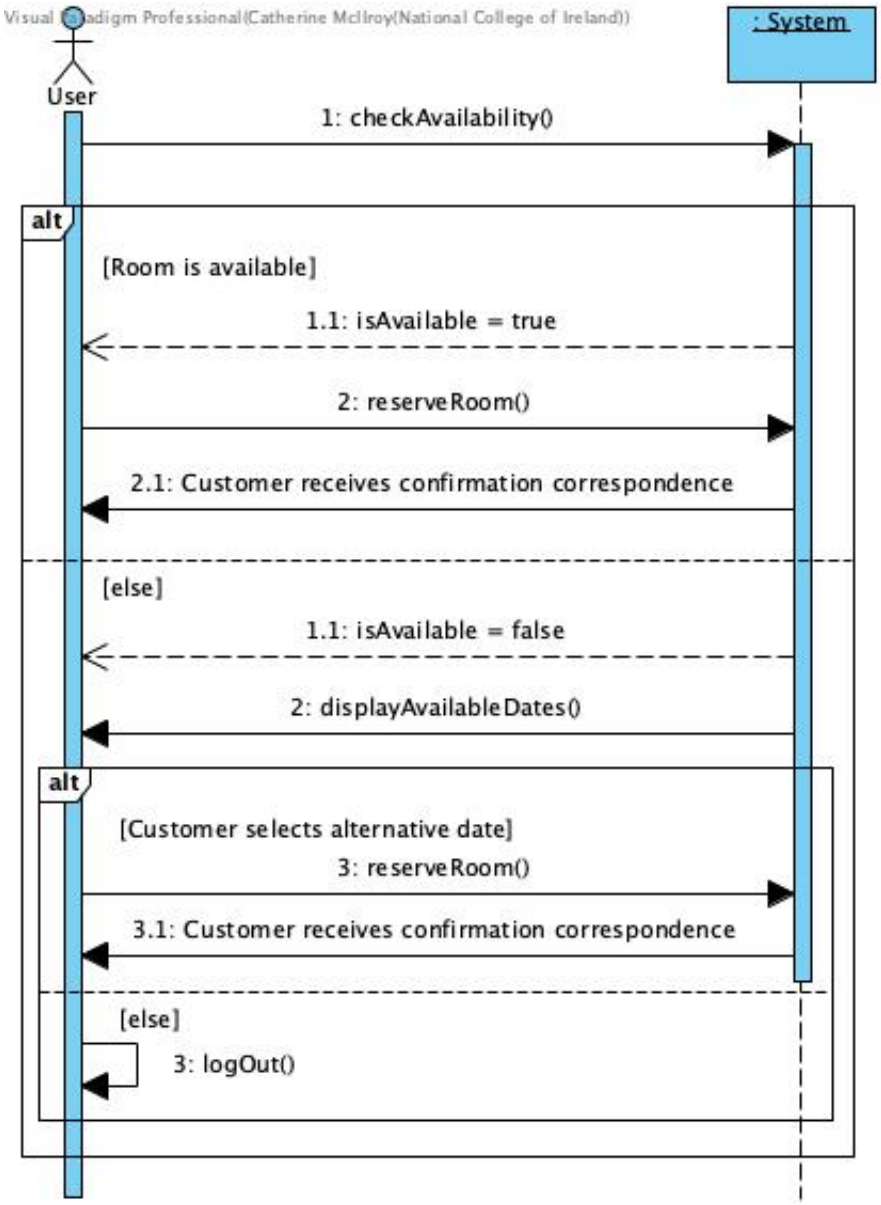
- **Receptionist class**
  - One receptionist can serve zero or many instances of Customer class, depending on how many customers need the receptionist to perform operations on their behalf.
  - One or many instances of Receptionist class can interact with one UserInterface.
- **Reservation class**
  - An instance of Reservation class is created by the CentralBookingSystem when a user reserves a room.
  - One Reservation object can be associated with one or many instances of the Room class.
  - Reservation class shows a composition relationship with Room class; a reservation cannot exist without at least one room.
  - When the boolean attribute isConfirmed is set to true, this triggers creation of an associated Invoice object.
- **Room class**
  - Shows a composition relationship with Hotel class. The room is part of the hotel and cannot exist without it.
- **Hotel class**
  - One instance of Hotel class can have one or many rooms.
  - One instance of Hotel class is associated with one instance of RoomInventory class.
  - The hotel must exist in order for the rooms or the room inventory to exist.

- **RoomInventory class**
  - Stores information relating to the number of rooms in the hotel and their current availability.
  - Shows a composition relationship with Hotel class. Without the hotel, there are no rooms and therefore no room inventory.
- **Manager class**
  - One instance of the Manager class can modify one instance of the RoomInventory class.

System Sequence Diagram (Task 6)

The System Sequence Diagram below shows the interactions involved in the “Make Reservation” use case previously covered in Task 3 (page 5).

This diagram showcases the interactions between the user and the System. In this case the System is represented as one whole object, and the internal workings of the System are not shown.



The process begins with the user checking room availability based on the dates of their stay, the number of guests, and the number of rooms they require.

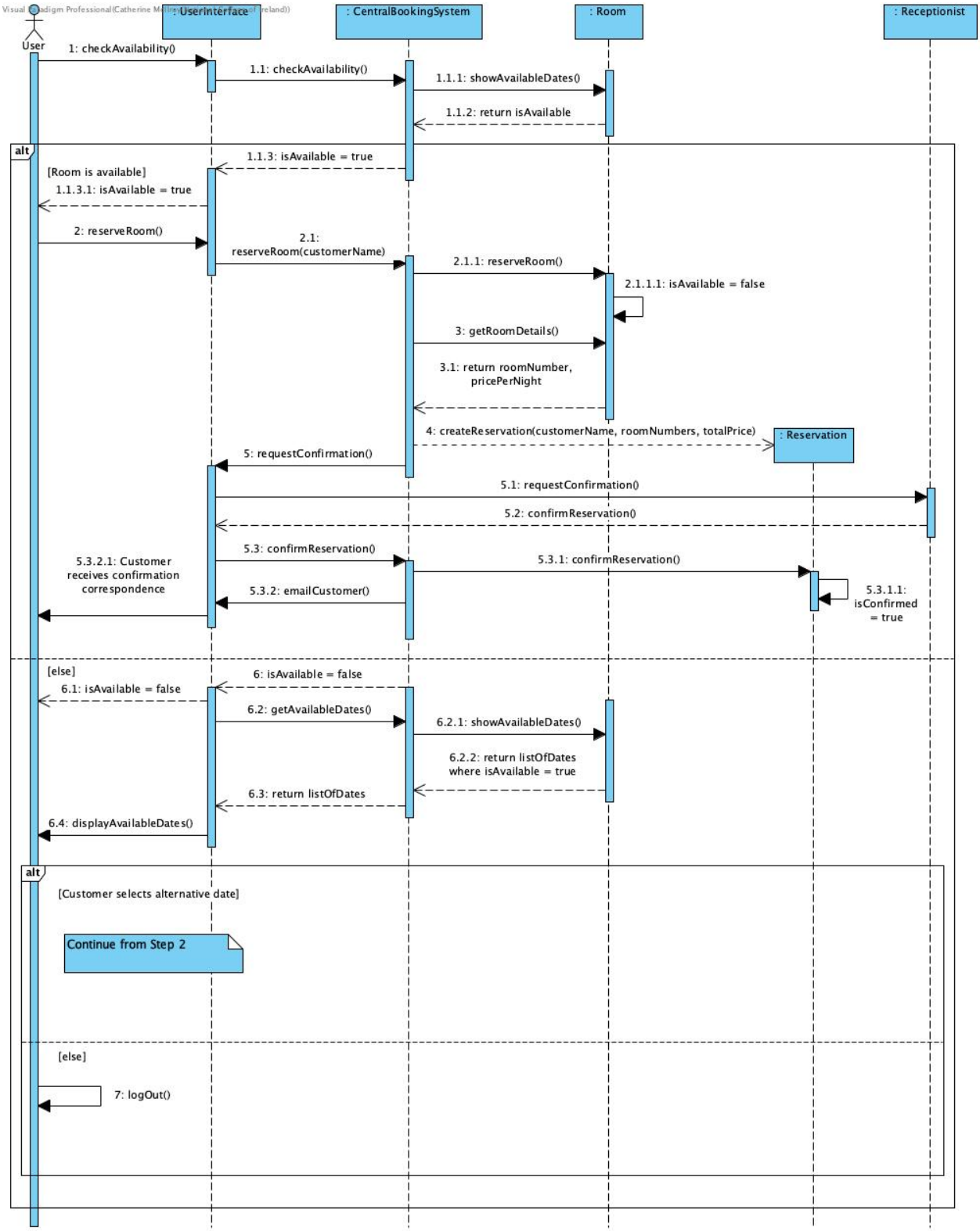
The System then returns a value of either true or false for room availability. If the room is available (i.e. isAvailable = true), the user then reserves the room, and the System sends confirmation correspondence to the customer.

If the room is not available on the selected date (i.e. isAvailable = false), the System returns a list of alternative available dates. The process can then take one of two paths: either the user selects an alternative date from this list, and the process continues as detailed above, or the user does not choose an alternative date, and logs out of the system without making a reservation.

A Sequence Diagram is also included on the following page along with a brief description.

Sequence Diagram

The Sequence Diagram differs from the System Sequence Diagram in that it shows the internal workings of the system, and the interactions between objects within the system. This diagram displays these events in chronological order.



The process again begins with the user checking room availability based on the dates of their stay, the number of guests, and the number of rooms they require. This action is performed via the UserInterface and triggers the checkAvailability() system operation.

The CentralBookingSystem requests available dates from the specified Room, which will return either isAvailable = true or false, depending on whether the room is available or already reserved.

The rest of the diagram is contained within nested alt frames in order to demonstrate the different flows.

- **Main Flow (Room is available)**

  - The boolean isAvailable attribute of the Room object returns true, indicating that the room is available to be reserved by the user.

  - The user then reserves the room, and the UserInterface object passes the customer name to the CentralBookingSystem object.

  - The CentralBookingSystem reserves the room, and the Room attribute isAvailable is modified to false.

  - The CentralBookingSystem requests the room details required to create a new Reservation.

  - A new instance of Reservation class is created, with the customerName, roomNumbers, and totalPrice attributes passed in as parameters. This results in the Reservation object being associated with the relevant Customer and Room(s), and provides the Reservation with the necessary information regarding total price, which is needed for an Invoice object to be generated.

  - The CentralBookingSystem requests confirmation of the reservation via the UserInterface.

  - Receptionist confirms the reservation via the UserInterface. This information is passed to the CentralBookingSystem and the boolean isConfirmed attribute of the Reservation object is set to true.

  - The CentralBookingSystem then emails confirmation and reservation details to the customer via the UserInterface.

- **Else/Alternate Flow (Room not available)**
  - The boolean isAvailable attribute of the Room object returns false, indicating that the room is unavailable to be reserved by the user on the dates specified.
  - The CentralBookingSystem retrieves from the Room a list of alternative dates currently available for reservation.
  - This list is returned to the UserInterface and displayed to the User.

At this point in the diagram, a second (nested) alt frame is used to show the two possible alternate flows of events.

- **Alternate Flow 1 (Customer selects alternative date)**
  - The user selects an alternative date, and the process continues from Step 2 (reserveRoom()) to Step 5.3.2.1 (Customer receives confirmation correspondence) inclusive.
- **Else/Alternate Flow 2 (Customer exits reservation system)**
  - The customer decides not to reserve the room on any of the alternative dates.
  - The user logs out of the reservation system without making a reservation.

Operation Contract (Task 7)

Operation	**reserveRoom( )**
Cross Reference	Use Cases: Make Reservation
Preconditions	The room is available
Postconditions	• A Reservation instance (res) was created. • res was associated with the current Room. • res was associated with the Customer. • Room.isAvailable was set to false.

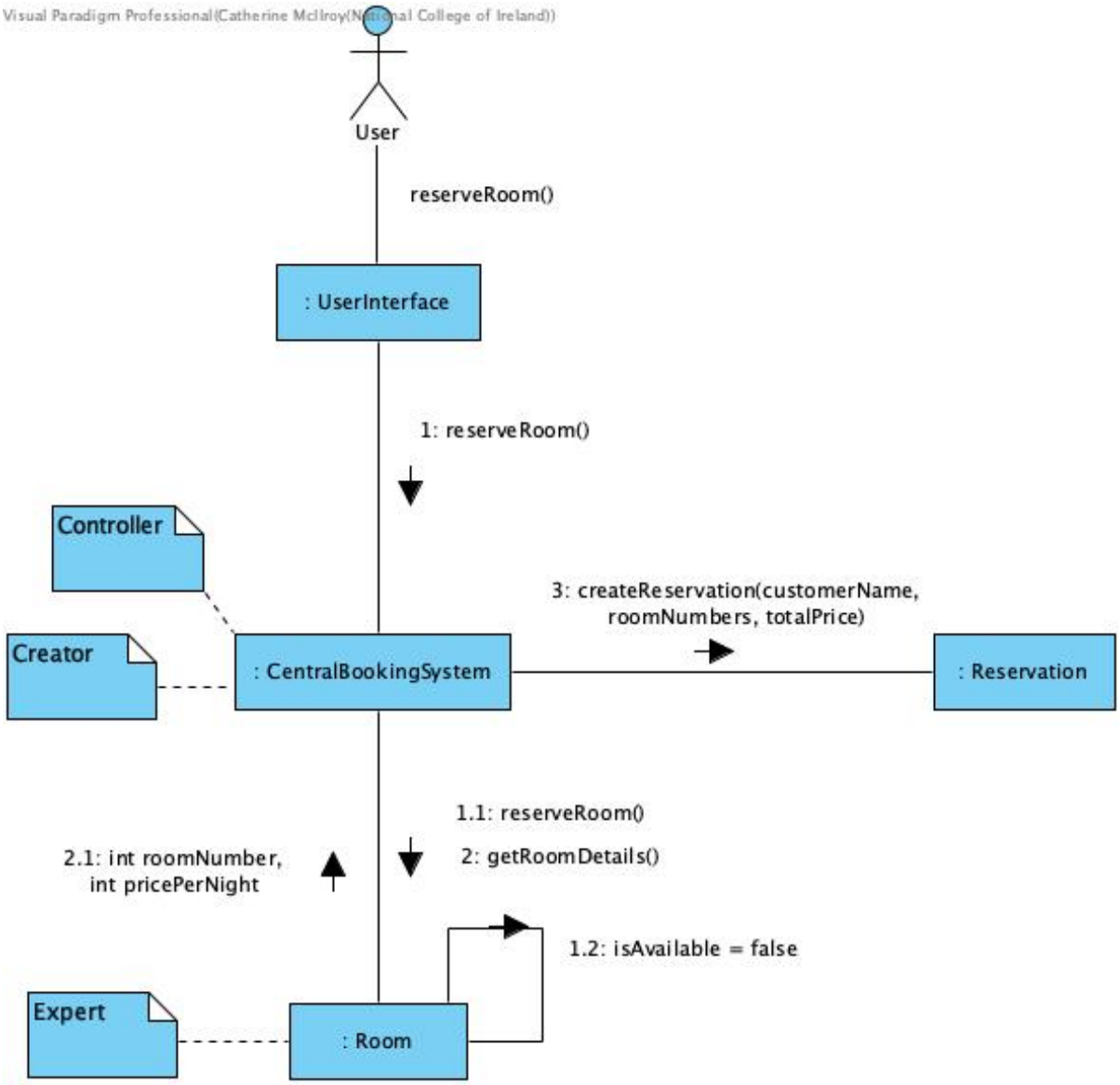
As can be seen from the above contract, prior to the reserveRoom() operation taking place, Room.isAvailable = true.

After execution of the reserveRoom() operation, a new instance of Reservation class, res, was created (instance creation).

This instance res was associated with the current instance of Room, and also associated with the Customer (associations formed).

The Room boolean attribute isAvailable became false (attribute modification).

Communication Diagram (Task 8)



The above communication diagram depicts the reserveRoom() operation previously described in Task 7 (page 16).

The user initiates the reserveRoom system event via the UserInterface, which triggers the reserveRoom() system operation.

The CentralBookingSystem follows the Controller design pattern in this case as it is a non-user interface object and is responsible for handling the incoming system events from the UserInterface class.

It also follows the Creator design pattern as it is responsible for creation of a new Reservation object.

The Room in this case shows the Expert design pattern as it holds the information required to complete the reserveRoom() operation and the information needed by the CentralBookingSystem for creation of the new Reservation.

Glossary (Task 5)

UML. Unified Modelling Language. A visual modelling language consisting of a standard set of diagrams which depict various aspects of a system's design.

Class. A blueprint for creating instances of that class. Defines the methods and variables to be inherited by objects of that class.

Instance. Represents an object of a class.

Actors. An external entity (outside of the system boundary) which interacts with the system.

Use Case. A distinct set of actions which are triggered by an actor and carried out within the system.

Attribute. A variable. Attributes of a class are inherited by every new object of that class.

Operation. A function, or method. Defines the behaviour of an object.

System Event. External input from an actor to the system. Triggers a corresponding system operation.

System Operation. An internal operation which is carried out in response to a system event.

References

1. GeeksforGeeks. (2017). *Unified Modeling Language (UML) | Sequence Diagrams*. [online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams.

2. www.oreilly.com. (n.d.). 11. Operation Contracts - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition [Book]. [online] Available at: https://learning.oreilly.com/library/view/applying-uml-and/0131489062/ch11.html#ch11lev1sec10 [Accessed 7 Mar. 2024].

3. www.ibm.com. (n.d.). *Communication diagrams*. [online] Available at: https://www.ibm.com/docs/en/rsm/7.5.0?topic=uml-communication-diagrams.

4. Wikipedia. (2023). *Sequence diagram*. [online] Available at: https://en.wikipedia.org/wiki/Sequence_diagram# [Accessed 5 Mar. 2024].

5. www.visual-paradigm.com. (n.d.). *How to Apply Numbering for Messages in Sequence Diagram?* [online] Available at: https://www.visual-paradigm.com/tutorials/numberingmessage.jsp#:~:text=Numbering%20(Single%20Level)- [Accessed 5 Mar. 2024].

6. Stack Overflow. (n.d.). *UML Design Patterns. Controller/Expert/Creator*. [online] Available at: https://stackoverflow.com/questions/48035561/uml-design-patterns-controller-expert-creator [Accessed 7 Mar. 2024].

7. Developer.com. (2004). *Using Design Patterns in UML*. [online] Available at: https://www.developer.com/design/using-design-patterns-in-uml/

8. Lucidchart. (n.d.). *UML Class Diagram Tutorial*. [online] Available at: https://www.lucidchart.com/pages/uml-class-diagram#:~:text=Popular%20among%20software%20engineers%20to [Accessed 5 Mar. 2024].

9. devcamp.com. (n.d.). *UML Class Diagram Associations, Multiplicity, and Navigability*. [online] Available at: https://devcamp.com/trails/uml-foundations/campsites/class-diagrams/guides/uml-class-diagram-associations-multiplicity-navigability.