S.E. LAB ASSIGNMENT

Expt. no: 3 | T.E. COMPS - A | Date: 28th Sept., 2022

Team:

1. Vyom Doshi (2020300009)
2. Shubham Golwal (2020300015)

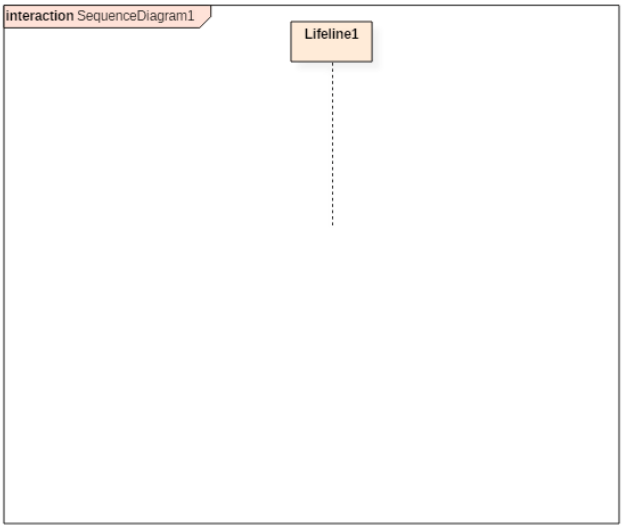
AIM: Design Sequence & Collaboration Diagram.

PROBLEM STATEMENT: The resort Property Management System is designed to manage managers with the ability to control their company's rental business. All aspects of short-term rental management are considered including reservations, accounting, maintenance, housekeeping, booking, and more.

THEORY:

What is an Interaction Diagram?

Interaction diagrams are used in UML to establish communication between objects. It does not manipulate the data associated with the particular communication path. Interaction diagrams mostly focus on message passing and how these messages make up one functionality of a system. Interaction diagrams are designed to display how the objects will realize the particular requirements of a system. The critical component in an interaction diagram is lifeline and messages.



Following are the different types of interaction diagrams defined in UML:

- Sequence diagram
- Collaboration diagram
- Timing diagram

What is a Sequence Diagram?

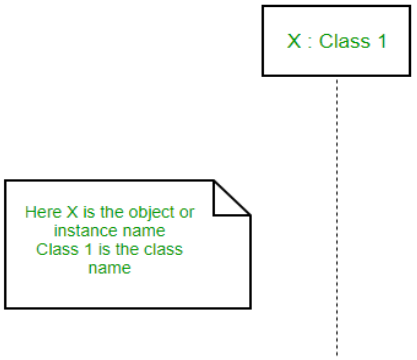
A Sequence Diagram simply depicts the interaction between objects in sequential order. The purpose of a sequence diagram in UML is to visualize the sequence of a message flow in the system. The sequence diagram shows the interaction between two lifelines as a time-ordered sequence of events.

Sequence Diagram Notations :

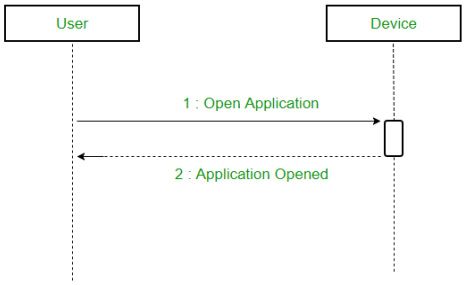
1. **Actors –** An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model

using the UML diagram.

2. **Lifelines –** A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram. The standard in UML for naming a lifeline follows the following format – Instance Name : Class Name



3. **Messages –** Communication between objects is depicted using messages. The messages appear in sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram. Messages can be broadly classified into the following categories :
   I. Synchronous messages – A synchronous message waits for a reply before the interaction can move forward. The sender waits until the receiver has completed the processing of the message. The caller continues only when it knows that the receiver has processed the previous message i.e. it receives a reply message. A large number of calls in object oriented programming are synchronous. We use a solid arrow head to represent a synchronous message.



   II. Asynchronous Messages – An asynchronous message does not wait for a reply from the receiver. The interaction moves forward irrespective of the receiver processing the previous message or not. We use a lined arrow head to represent an asynchronous message.
   III. Create message – We use a Create message to instantiate a new object in the sequence diagram. There are situations when a particular message call requires the creation of an object. It is represented with a dotted arrow and create word labelled on it to specify that it is the create Message symbol. For example – The creation of a new order on a e-commerce website would require a new object of Order class to be created.
   IV. Delete Message – We use a Delete Message to delete an object. When an object is deallocated memory or is destroyed within the system we use the Delete Message symbol. It destroys the

occurrence of the object in the system.It is represented by an arrow terminating with a x. For example – In the scenario below when the order is received by the user, the object of order class can be destroyed.

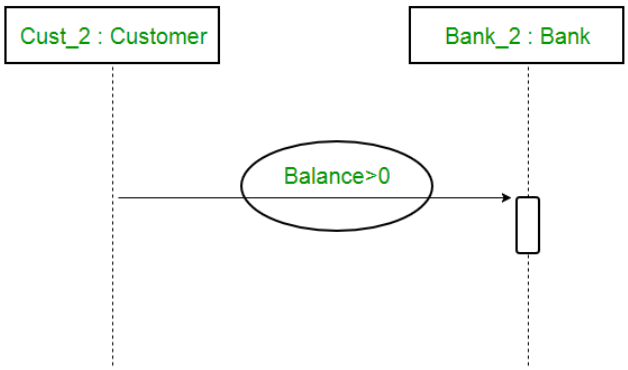
V. Self Message – Certain scenarios might arise where the object needs to send a message to itself. Such messages are called Self Messages and are represented with a U shaped arrow. Figure – self messageFor example – Consider a scenario where the device wants to access its webcam. Such a scenario is represented using a self message.

VI. Reply Message – Reply messages are used to show the message being sent from the receiver to the sender. We represent a return/reply message using an open arrowhead with a dotted line. The interaction moves forward only when a reply message is sent by the receiver.

VII. Found Message – A Found message is used to represent a scenario where an unknown source sends the message. It is represented using an arrow directed towards a lifeline from an end point.

VIII. Lost Message – A Lost message is used to represent a scenario where the recipient is not known to the system. It is represented using an arrow directed towards an end point from a lifeline. For example: Consider a scenario where a warning is generated.

Guards – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process. For example: In order to be able to withdraw cash, having a balance greater than zero is a condition that must be met as shown below.



Benefits of a Sequence Diagram

- Sequence diagrams are used to explore any real application or a system.
- Sequence diagrams are used to represent message flow from one object to another object.
- Sequence diagrams are easier to maintain.
- Sequence diagrams are easier to generate.
- Sequence diagrams can be easily updated according to the changes within a system.
- Sequence diagram allows reverse as well as forward engineering.

Drawbacks of a sequence diagram

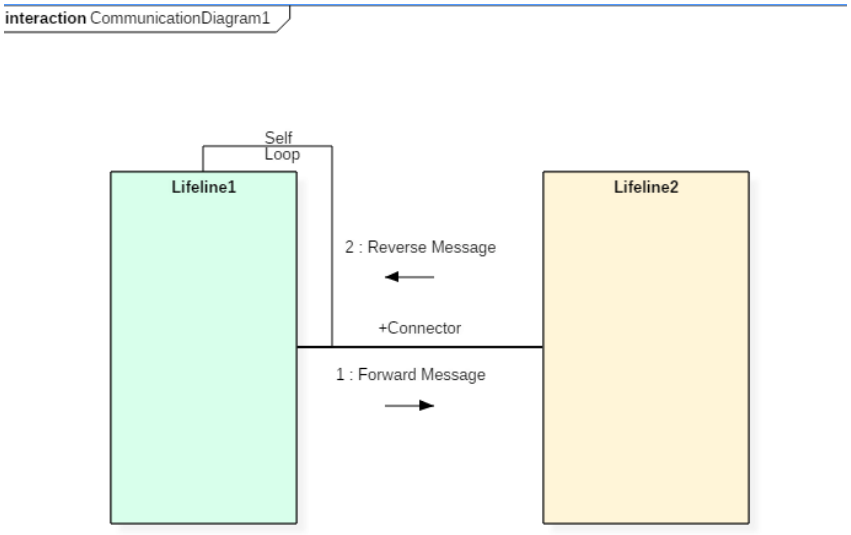
- Sequence diagrams can become complex when too many lifelines are involved in the system.
- If the order of message sequence is changed, then incorrect results are produced.
- Each sequence needs to be represented using different message notations, which can be a little complex.
- The type of message decides the type of sequence inside the diagram.

What is the Collaboration Diagram?

The collaboration Diagram depicts the relationships and interactions among software objects. They are used to understand the object architecture within a system rather than the flow of a message as in a sequence diagram. They are also known as “Communication Diagrams.”

Benefits of Collaboration Diagram

- It is also called as a communication diagram.
- It emphasizes the structural aspects of an interaction diagram – how lifeline connects.
- Its syntax is similar to that of sequence diagram except that lifeline don't have tails.
- Messages passed over sequencing is indicated by numbering each message hierarchically.
- Compared to the sequence diagram communication diagram is semantically weak.
- Object diagrams are special case of communication diagram.
- It allows you to focus on the elements rather than focusing on the message flow as described in the sequence diagram.
- Sequence diagrams can be easily converted into a collaboration diagram as collaboration diagrams are not very expressive.
- While modeling collaboration diagrams w.r.t sequence diagrams, some information may be lost.

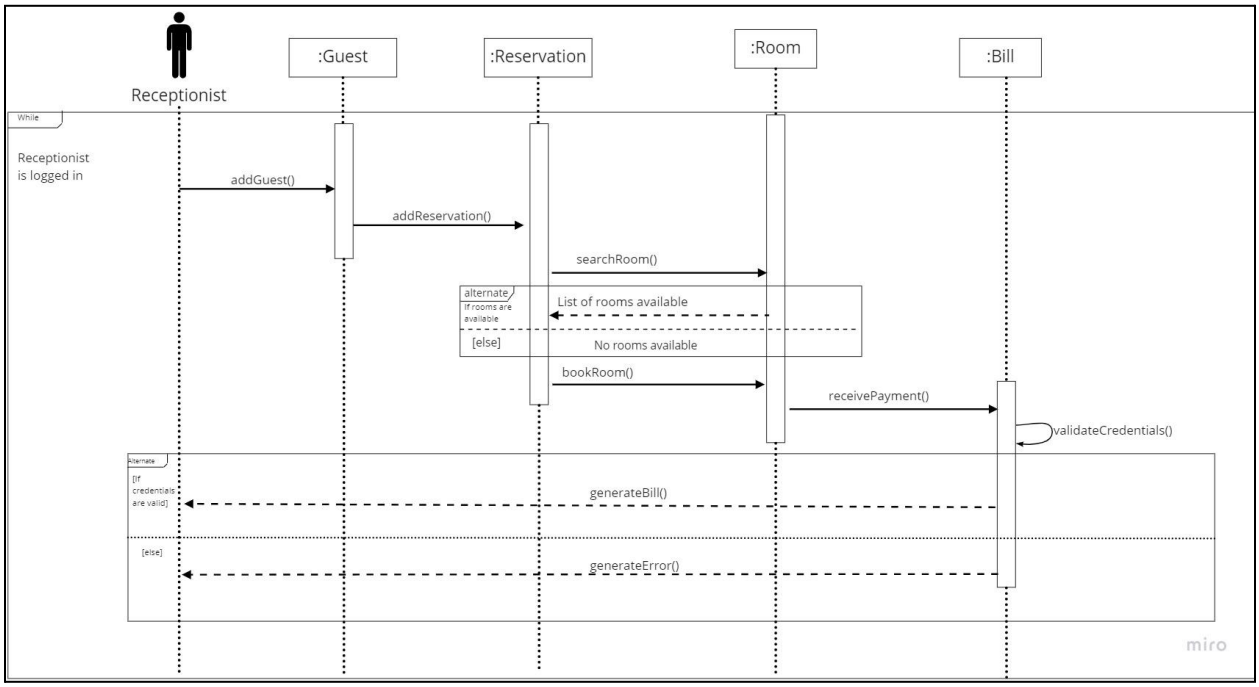


Drawbacks of a Collaboration Diagram

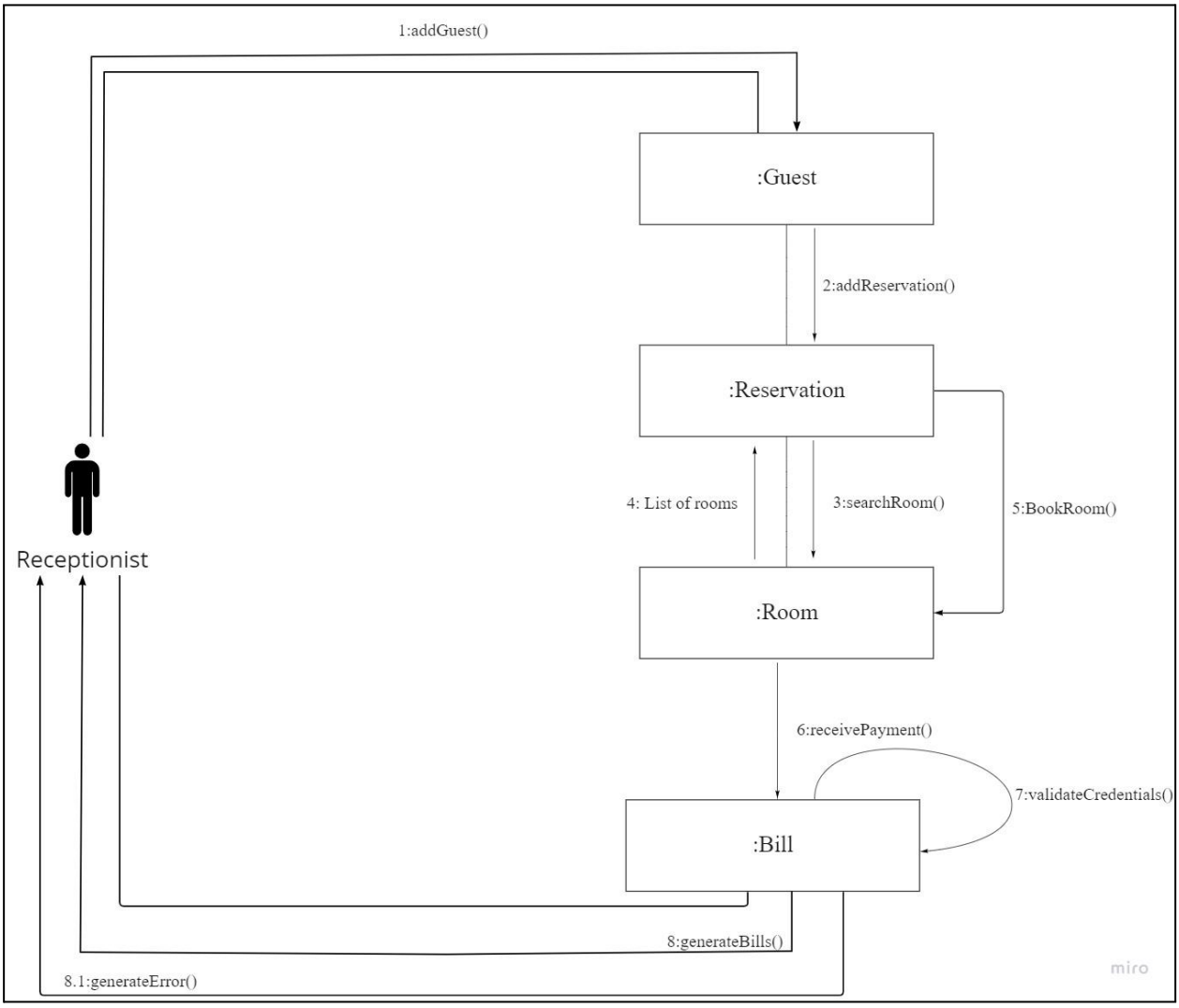
- Collaboration diagrams can become complex when too many objects are present within the system.
- It is hard to explore each object inside the system.
- Collaboration diagrams are time-consuming.
- The object is destroyed after the termination of a program.
- The state of an object changes momentarily, which makes it difficult to keep track of every single change that occurs within an object of a system.

IMPLEMENTATION:

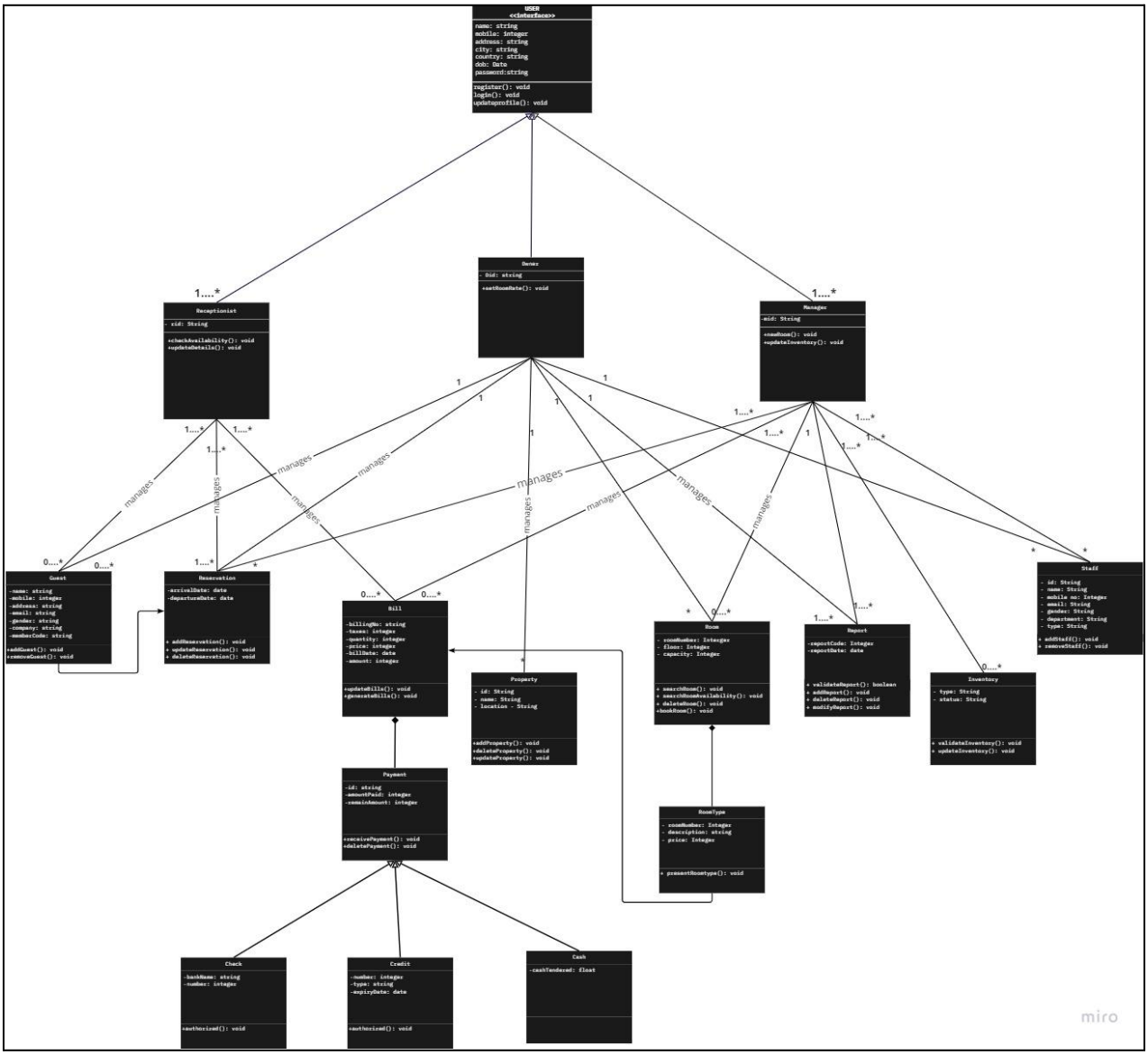
1. Sequence Diagram:



2. Collaboration Diagram:



3. **Class Diagram (Just for Reference)**



CONCLUSION:

Successfully implemented the sequence diagram and collaboration diagram concepts to make them for our Management System. This helped us understand the concepts of sequence diagrams and collaboration diagrams, where they are used, and their advantages and limitations. We also realized how much intricacies and clarity of thoughts matter here, especially clarity in the class diagram.

REFERENCES:

https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
https://www.javatpoint.com/uml-collaboration-diagram