Names:
Vishal Salvi (2019230069)
Shivam Pawar (2019230068)
Shreyas Patel (2018130037)
Batch: C
Class: TE comps

Experiment No 7

Aim: To create Sequence and State Diagram for Mumbai Tour and guide management System.

Theory:
Interaction Diagram:

What is Interaction diagram?

INTERACTION DIAGRAM is 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.

Various UML elements typically own interaction diagrams. The details of interaction can be shown using several notations such as sequence diagram, timing diagram, communication/collaboration diagram. Interaction diagrams capture the dynamic behaviour of any system.

Purpose of an Interaction Diagram

Interaction diagrams help you to visualize the interactive behaviour of a system. Interaction diagrams are used to represent how one or more objects in the system connect and communicate with each other.

Interaction diagrams focus on the dynamic behaviour of a system. An interaction diagram provides us the context of an interaction between one or more lifelines in the system.

In UML, the interaction diagrams are used for the following purposes:

- Interaction diagrams are used to observe the dynamic behaviour of a system.
- Interaction diagram visualizes the communication and sequence of message passing in the system.
- Interaction diagram represents the structural aspects of various objects in the system.
- Interaction diagram represents the ordered sequence of interactions within a system.
- Interaction diagram provides the means of visualizing the real time data via UML.
- Interaction diagrams can be used to explain the architecture of an object-oriented or a distributed system.

Important terminology

An interaction diagram contains lifelines, messages, operators, state invariants and constraints.

Lifeline

A lifeline represents a single participant in an interaction. It describes how an instance of a specific classifier participates in the interaction.

A lifeline represents a role that an instance of the classifier may play in the interaction. Following are various attributes of a lifeline,

1. Name
    1. It is used to refer the lifeline within a specific interaction.
    2. A name of a lifeline is optional.
2. Type
    1. It is the name of a classifier of which the lifeline represents an instance.
3. Selector
    1. It is a Boolean condition which is used to select a particular instance that satisfies the requirement.
    2. Selector attribute is also optional.

The notation of lifeline is explained in the notation section.

Messages

A message is a specific type of communication between two lifelines in an interaction. A message involves following activities,

1. A call message which is used to call an operation.
2. A message to create an instance.
3. A message to destroy an instance.
4. For sending a signal.

When a lifeline receives a call message, it acts as a request to invoke an operation that has a similar signature as specified in the message. When a lifeline is executing a message, it has a focus of control. As the interaction progresses over time, the focus of control moves between various lifelines. This movement is called a flow of control.

Following are the messages used in an interaction diagram:

Message Name	Meaning
Synchronous message	The sender of a message keeps waiting for the receiver to return control from the message execution.
Asynchronous message	The sender does not wait for a return from the receiver; instead, it continues the execution of a next message.
Return message	The receiver of an earlier message returns the focus of control to the sender.
Object creation	The sender creates an instance of a classifier.
Object destruction	The sender destroys the created instance.
Found message	The sender of the message is outside the scope of interaction.
Lost message	The message never reaches the destination, and it is lost in the interaction.

State invariants and constraints

When an instance or a lifeline receives a message, it can cause it to change the state. A state is a condition or a situation during a lifetime of an object at which it satisfies some constraint, performs some operations, and waits for some event.

In interaction diagram, not all messages cause to change the state of an instance. Some messages do not the values of some attribute. It has no side effects on the state of an object.

Operator

An operator specifies an operation on how the operands are going to be executed. The operators in UML supports operations on data in the form of branching as well as an iteration. Various operators can be used to ensure the use of iteration and branching in the UML model. The opt and alt operators are used for branching operations. The loop operator is used to ensure the iteration operations in which a condition is executed repeatedly until the satisfying result is produced. Break operator is used inside the loop or iteration operations. It ensures that the loop is terminated whenever a break operator is encountered. If a break condition is not specified, then the loop executes the infinite number of times, which results in crashing the program.

Following are the operators used in an interaction diagram:

Operator	Name	Meaning
Opt	Option	An operand is executed if the condition is true. e.g., If else
Alt	Alternative	The operand, whose condition is true, is executed. e.g., switch
Loop	Loop	It is used to loop an instruction for a specified period.
Break	Break	It breaks the loop if a condition is true or false, and the next instruction is executed.
Ref	Reference	It is used to refer to another interaction.
Par	Parallel	All operands are executed in parallel.

Iteration

In an interaction diagram, we can also show iteration using an iteration expression. An iteration expression consists of an iteration specifier and an optional iteration clause. There is no pre-specified syntax for UML iteration.

In iteration to show that messages are being sent in parallel, parallel iteration specifier is used. A parallel iteration specifier is denoted by *//. Iteration in UML is achieved by using the loop operator.

Branching

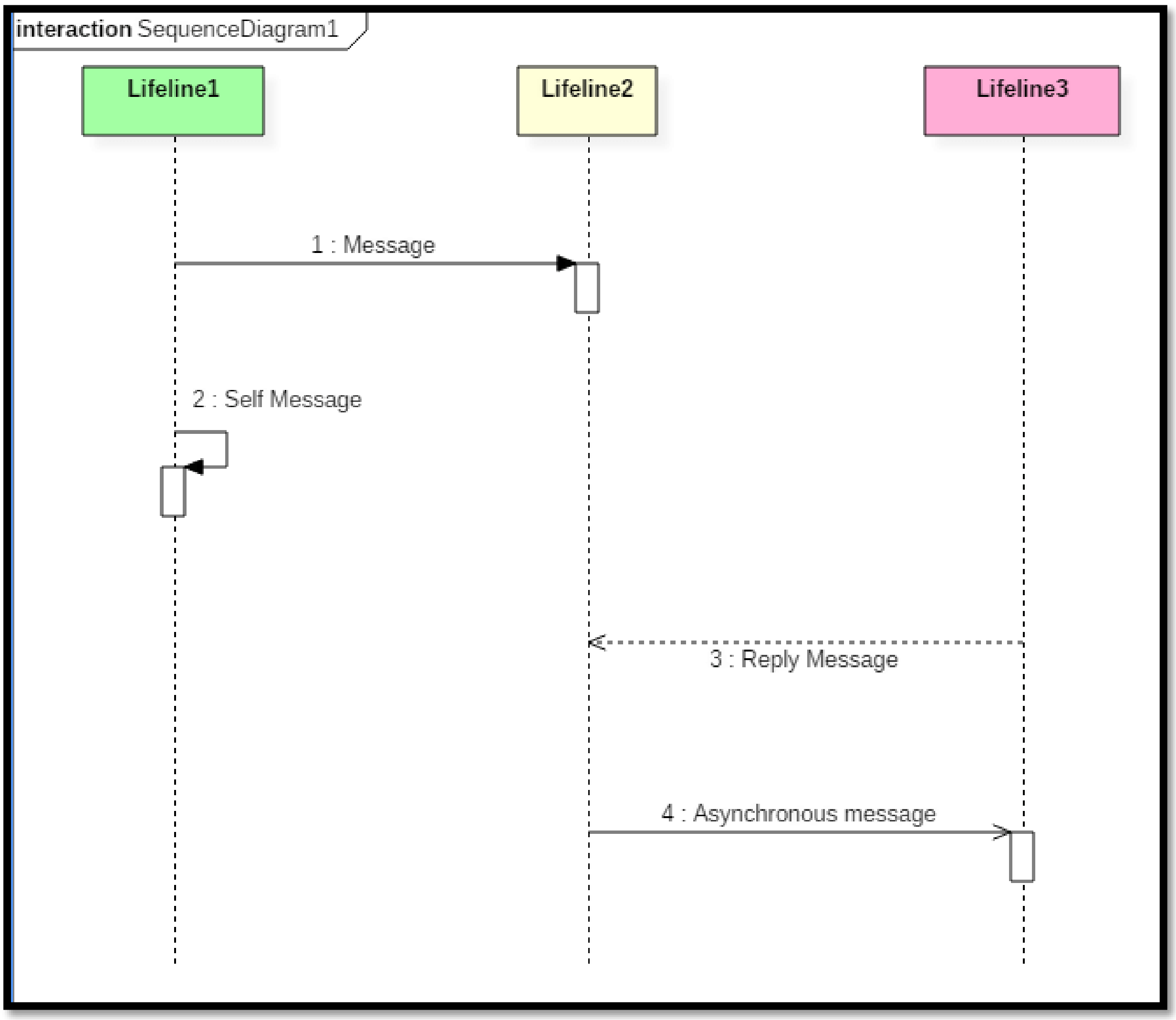
In an interaction diagram, we can represent branching by adding guard conditions to the messages. Guard conditions are used to check if a message can be sent forward or not. A message is sent forward only when its guard condition is true. A message can have multiple guard conditions, or multiple messages can have the same guard condition. Branching in UML is achieved with the help of alt and opt, operators.

These are some of the **most important** terminologies used in UML interaction diagram.

What is a Sequence Diagram?

A **SEQUENCE DIAGRAM** simply depicts interaction between objects in a 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.

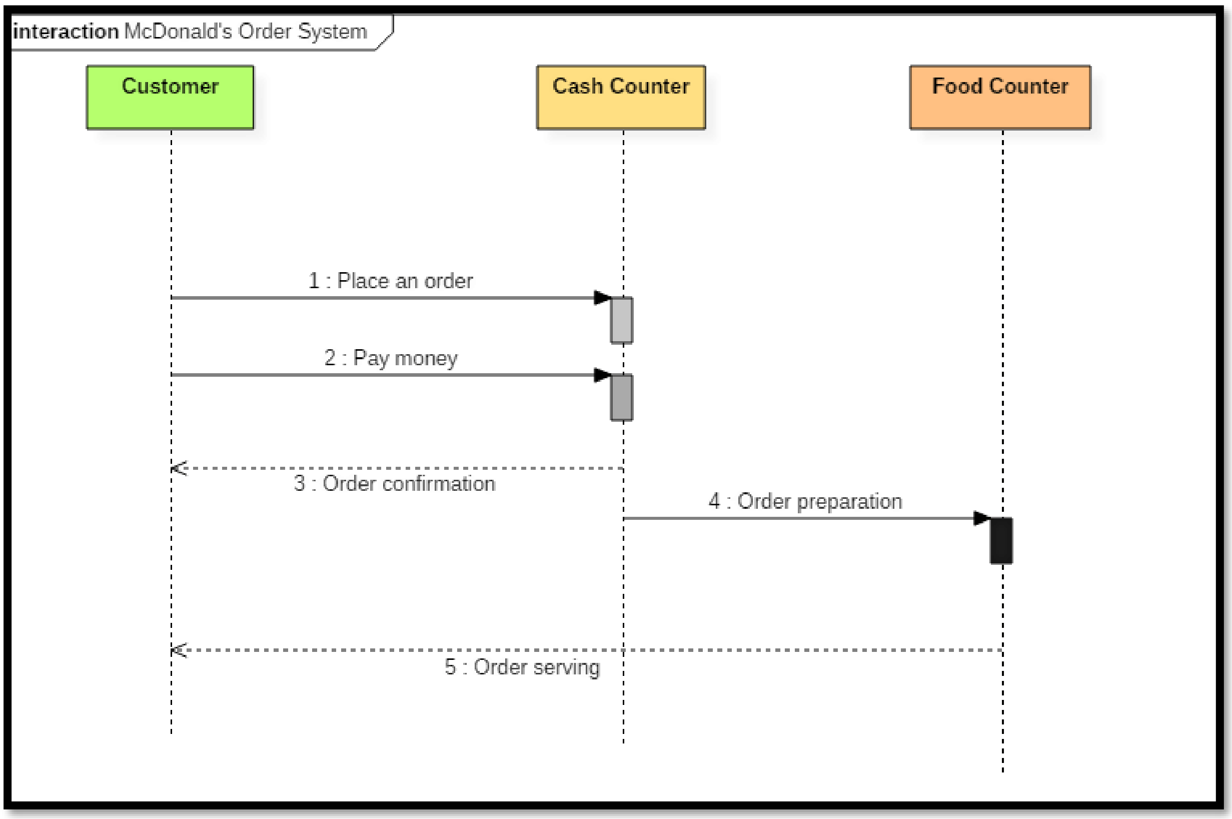
- A sequence diagram shows an implementation of a scenario in the system. Lifelines in the system take part during the execution of a system.
- In a sequence diagram, a lifeline is represented by a vertical bar.
- A message flow between two or more objects is represented using a vertical dotted line which extends across the bottom of the page.
- In a sequence diagram, different types of messages and operators are used which are described above.
- In a sequence diagram, iteration and branching are also used.



The above sequence diagram contains lifeline notations and notation of various messages used in a sequence diagram such as a create, reply, asynchronous message, etc.

Sequence diagram example

The following sequence diagram example represents McDonald's ordering system:



Sequence diagram of Mcdonald's ordering system

The ordered sequence of events in a given sequence diagram is as follows:

1. Place an order.
2. Pay money to the cash counter.
3. Order Confirmation.
4. Order preparation.
5. Order serving.

If one changes the order of the operations, then it may result in crashing the program. It can also lead to generating incorrect or buggy results. Each sequence in the above-given sequence diagram is denoted using a different type of message. One cannot use the same type of message to denote all the interactions in the diagram because it creates complications in the system.

You must be careful while selecting the notation of a message for any particular interaction. The notation must match with the particular sequence inside the diagram.

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 notation, which can be a little complex.
- The type of message decides the type of sequence inside the diagram.

What is a State Diagram?

STATE DIAGRAM is used to capture the behaviour of a software system. UML State machine diagrams can be used to model the behaviour of a class, a subsystem, a package, or even an entire system. It is also called a State chart or State Transition diagram.

State chart diagrams provide us an efficient way to model the interactions or communication that occur within the external entities and a system. These diagrams are used to model the event-based system. A state of an object is controlled with the help of an event.

State chart diagrams are used to describe various states of an entity within the application system.

How to draw a State chart diagram?

State chart diagrams are used to describe the various state that an object passes through. A transition between one state into another state occurs because of some triggered event. To draw a state diagram, one must identify all the possible states of any particular entity.

The purpose of these UML diagrams is to represent states of a system. States plays a vital role in state transition diagrams. All the essential object, states, and the events that cause changes within the states must be analysed first before implementing the diagram.

Following rules must be considered while drawing a state chart diagram:

1. The name of a state transition must be unique.
2. The name of a state must be easily understandable and describe the behaviour of a state.
3. If there are multiple objects, then only essential objects should be implemented.
4. Proper names for each transition and an event must be given.

When to use State Diagrams?

State diagrams are used to implement real-life working models and object-oriented systems in depth. These diagrams are used to compare the dynamic and static nature of a system by capturing the dynamic behavior of a system.

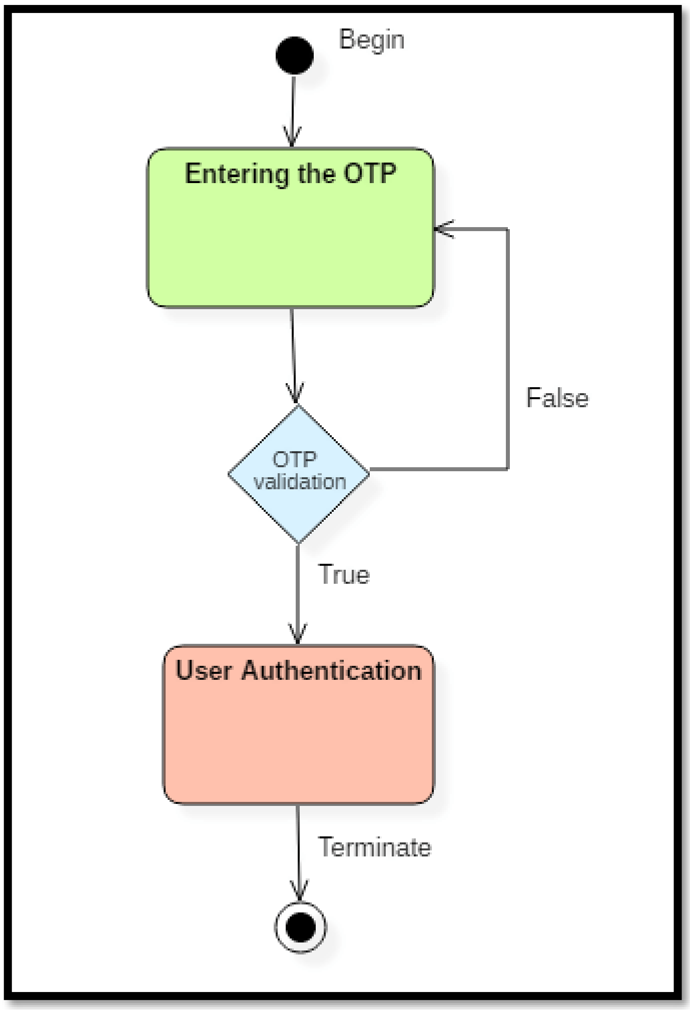
Statechart diagrams are used to capture the changes in various entities of the system from start to end. They are used to analyze how an event can trigger change within multiple states of a system.

State char diagrams are used,

1. To model objects of a system.
2. To model and implement interactive systems.
3. To display events that trigger changes within the states.

Example of State Machine

Following state chart diagram represents the user authentication process.



UML state diagram

There are a total of two states, and the first state indicates that the OTP has to be entered first. After that, OTP is checked in the decision box, if it is correct, then only state transition will occur, and the user will be validated. If OTP is incorrect, then the transition will not take place, and it will again go back to the beginning state until the user enters the correct OTP.

Why State Machine Diagram?

Statechart diagram is used to capture the dynamic aspect of a system. State machine diagrams are used to represent the behaviour of an application. An object goes through various states during its lifespan. The lifespan of an object remains until the program is terminated. The object goes from multiple states depending upon the event that occurs within the object. Each state represents some unique information about the object.

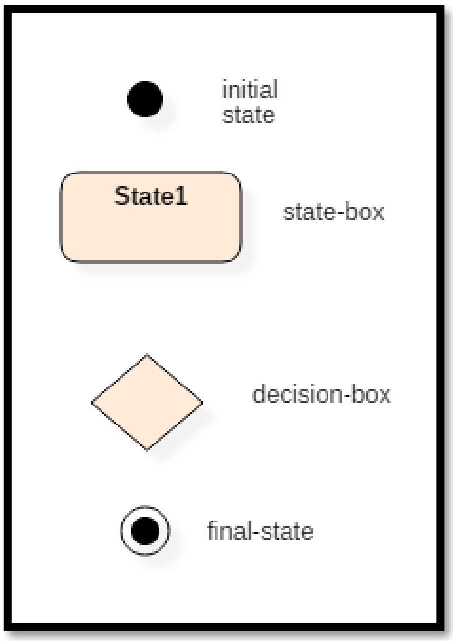
Statechart diagrams are used to design interactive systems that respond to either internal or external event. Statechart diagram visualizes the flow of execution from one state to another state of an object.

It represents the state of an object from the creation of an object until the object is destroyed or terminated.

The primary purpose of a statechart diagram is to model interactive systems and define each and every state of an object. Statechart diagrams are designed to capture the dynamic behaviour of an application system. These diagrams are used to represent various states of a system and entities within the system.

Notation and Symbol for State Machine

Following are the various notations that are used throughout the state chart diagram. All these notations, when combined, make up a single diagram.



UML state diagram notations

Initial state

The initial state symbol is used to indicate the beginning of a state machine diagram.

Final state

This symbol is used to indicate the end of a state machine diagram.

Decision box

It contains a condition. Depending upon the result of an evaluated guard condition, a new path is taken for program execution.

Transition

A transition is a change in one state into another state which is occurred because of some event. A transition causes a change in the state of an object.

State box

It is a specific moment in the lifespan of an object. It is defined using some condition or a statement within the classifier body. It is used to represent any static as well as dynamic situations.

It is denoted using a rectangle with round corners. The name of a state is written inside the rounded rectangle.

The name of a state can also be placed outside the rectangle. This can be done in case of composite or submachine states. One can either place the name of a state within the rectangle or outside the rectangle in a tabular box. One cannot perform both at the same time.

A state can be either active or inactive. When a state is in the working mode, it is active, as soon as it stops executing and transits into another state, the previous state becomes inactive, and the current state becomes active.

How to draw a Interaction diagram?

Interaction diagrams are used to represent the interactive behaviour of a system. Interaction diagrams focus on the dynamic behaviour of a system. An interaction diagram provides us the context of an interaction between one or more lifelines in the system.

To draw an interaction diagram, you have first to determine the scenario for which you have to draw an interaction diagram. After deciding the situation, identify various lifelines that are going to be involved in the interaction. Categorize all the lifeline elements and explore them to identify possible connections and how the lifelines are related to one another. To draw an interaction diagram, the following things are required:

1. The total number of lifelines that are going to be part of an interaction
2. is a sequence of message flow within various objects of a system.

3. Various operators to ease the functionality of an interaction diagram.
4. Various types of messages to display the interaction more clearly and in a precise manner.
5. The ordered sequence of messages.
6. Organization and a structure of an object.
7. Various time constructs of an object.

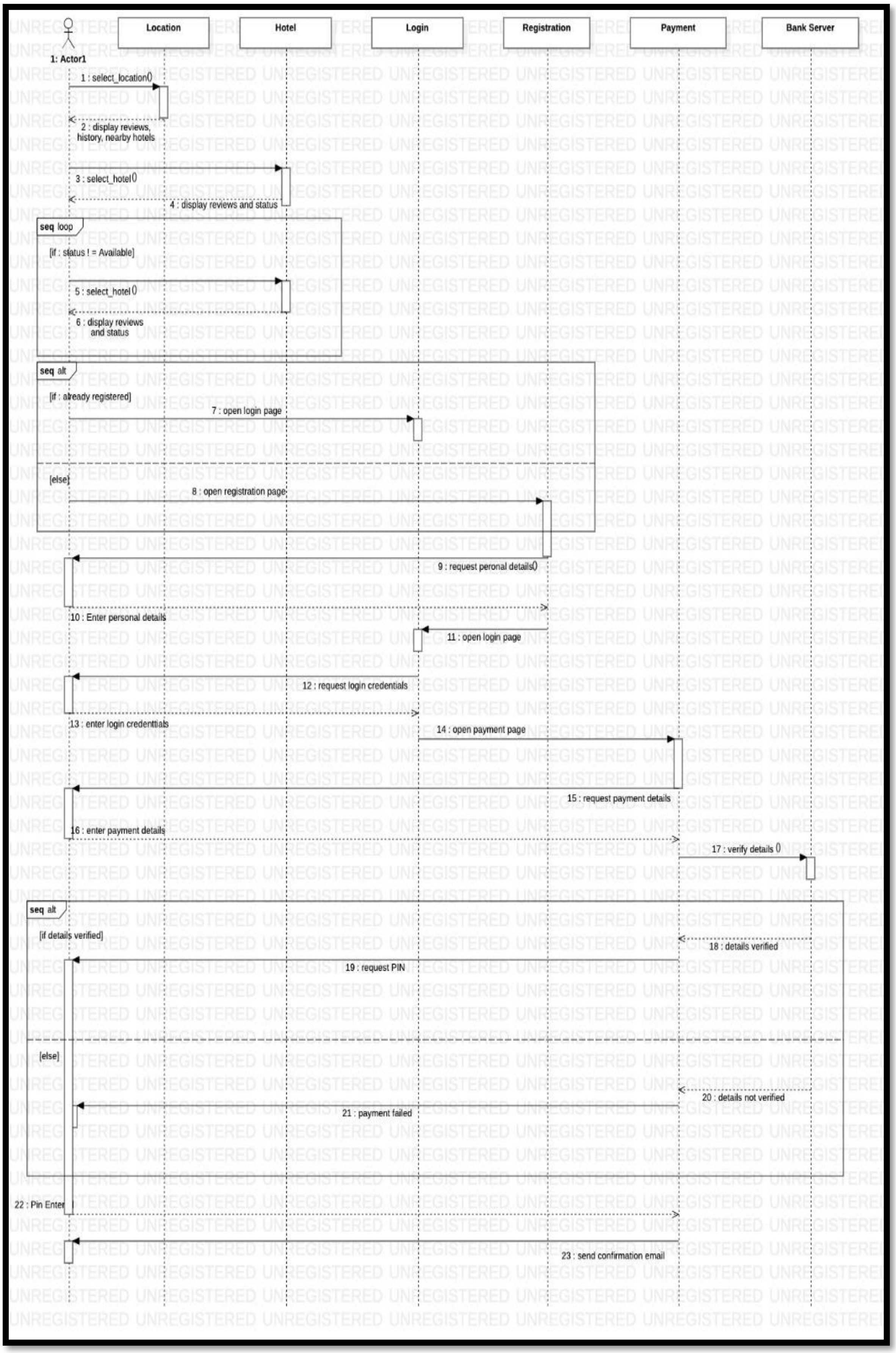
Use of an interaction diagram

Interaction diagrams consist of a sequence diagram, collaboration diagram, and timing diagrams. Following is the specific purpose of an interaction diagram:

- Sequence diagrams are used to explore any real application or a system.
- Interaction diagrams are used to explore and compare the use of sequence, collaborations, and timing diagrams.
- Interaction diagrams are used to capture the behaviour of a system. It displays the dynamic structure of a system.
- Sequence diagrams are used to represent message flow from one object to another object.
- Collaboration diagrams are used to understand the object architecture of a system rather than message flow.
- Interaction diagrams are used to model a system as a time-ordered sequence of events.
- Interaction diagrams are used in reverse as well as forward engineering.
- Interaction diagrams are used to organize the structure of interactive elements.

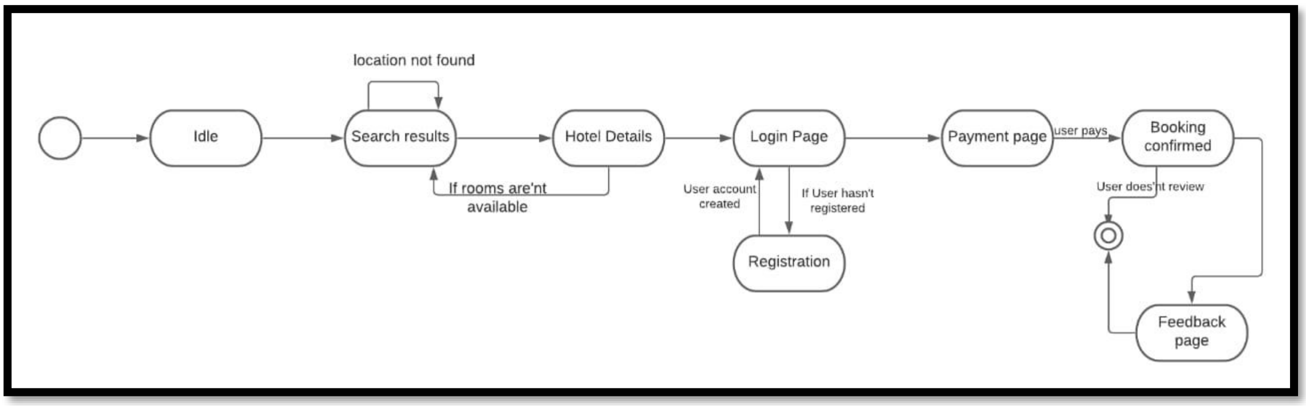
Diagram:

Sequence Diagram:



1: Actor1
Location
Hotel
Login
Registration
Payment
Bank Server
1 : select_location()
2 : display reviews, history, nearby hotels
3 : select_hotel()
4 : display reviews and status
seq loop
[if : status ! = Available]
5 : select_hotel()
6 : display reviews and status
seq alt
[if : already registered]
7 : open login page
[else]
8 : open registration page
9 : request peronal details()
10 : Enter personal details
11 : open login page
12 : request login credentials
13 : enter login credenttials
14 : open payment page
15 : request payment details
16 : enter payment details
17 : verify details ()
seq alt
[if details verified]
18 : details verified
19 : request PIN
[else]
20 : details not verified
21 : payment failed
22 : Pin Enter
23 : send confirmation email
UNREGISTERED

State Diagram:



Conclusion:

Thus, from this experiment we learn Interaction diagram and draw Sequence and state diagram.