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

Experiment No 3

Aim: To create Use case Diagram and Use case Description for Mumbai Tour and guide management System.

Theory:

What is the Use Case Diagram?

Use Case Diagram captures the system's functionality and requirements by using actors and use cases. Use Cases model the services, tasks, function that a system needs to perform. Use cases represent high-level functionalities and how a user will handle the system. Use-cases are the core concepts of Unified Modelling language modeling.

Why Use-Case diagram?

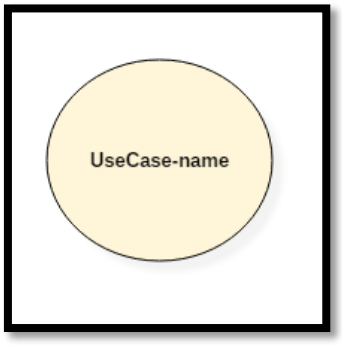
A Use Case consists of use cases, persons, or various things that are invoking the features called as actors and the elements that are responsible for implementing the use cases. Use case diagrams capture the dynamic behaviour of a live system. It models how an external entity interacts with the system to make it work. Use case diagrams are responsible for visualizing the external things that interact with the part of the system.

Use-case diagram notations

Following are the common notations used in a use case diagram:

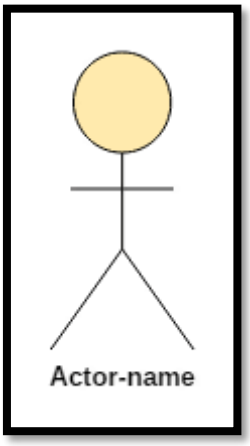
Use-case:

Use cases are used to represent high-level functionalities and how the user will handle the system. A use case represents a distinct functionality of a system, a component, a package, or a class. It is denoted by an oval shape with the name of a use case written inside the oval shape. The notation of a use case in UML is given below:

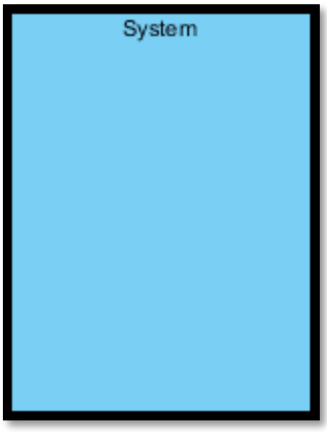


Actor:

It is used inside use case diagrams. The actor is an entity that interacts with the system. A user is the best example of an actor. An actor is an entity that initiates the use case from outside the scope of a use case. It can be any element that can trigger an interaction with the use case. One actor can be associated with multiple use cases in the system. The actor notation in UML is given below.

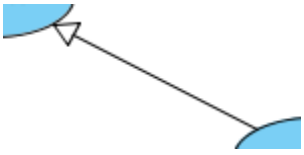


System boundary boxes: A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system.



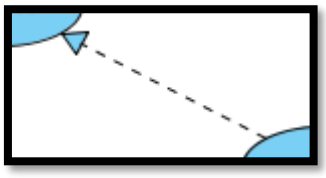
Packages: A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.

Generalization



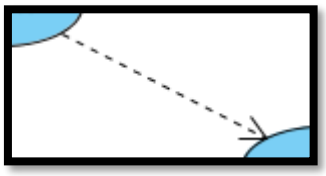
A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element share the same specification with. the more general the model element but carries more details in extra.

Realization



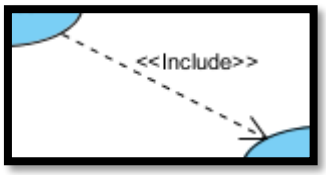
A realization is a relationship between a specification and its implementation.

Dependency



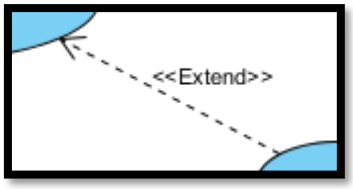
A dependency relationship represents that a model element relies on another model element for specification and/or implementation.

Include



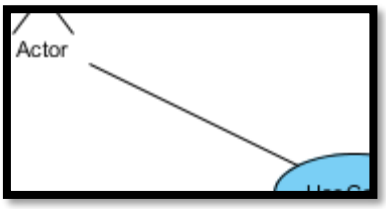
An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

Extend



An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.

Association



Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

How to draw a use-case diagram?

To draw a use case diagram in UML first one need to analyse the entire system carefully. You have to find out every single function that is provided by the system. After all the functionalities of a system are found out, then these functionalities are converted into various use cases which will be used in the use case diagram.

A use case is nothing but a core functionality of any working system. After organizing the use cases, we have to enlist the various actors or things that are going to interact with the system. These actors are responsible for invoking the functionality of a system. Actors can be a person or a thing. It can also be a private entity of a system. These actors must be relevant to the functionality or a system they are interacting with.

After the actors and use cases are enlisted, then you have to explore the relationship of a particular actor with the use case or a system. One must identify the total number of ways an actor could interact with the system. A single actor can interact with multiple use cases at the same time, or it can interact with numerous use cases simultaneously.

Following rules must be followed while drawing use-case for any system:

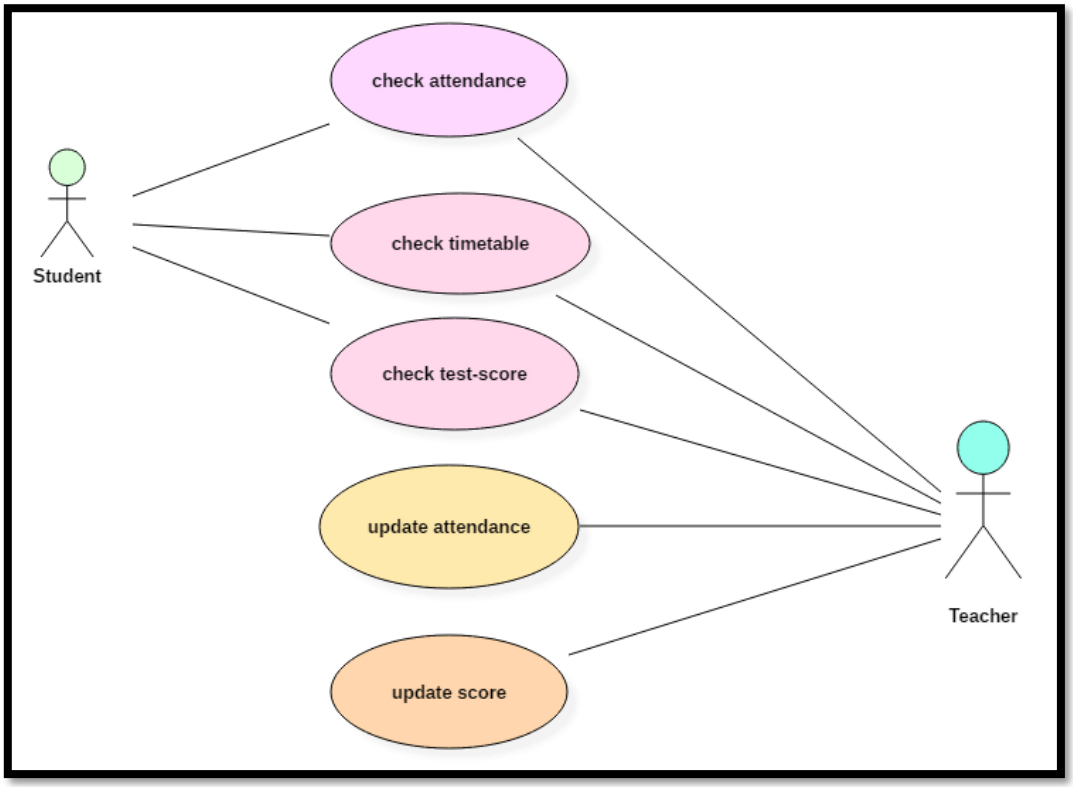
1. The name of an actor or a use case must be meaningful and relevant to the system.
2. Interaction of an actor with the use case must be defined clearly and in an understandable way.
3. Annotations must be used wherever they are required.
4. If a use case or an actor has multiple relationships, then only significant interactions must be displayed.

Tips for drawing a use-case diagram

1. A use case diagram should be as simple as possible.
2. A use case diagram should be complete.
3. A use case diagram should represent all interactions with the use case.
4. If there are too many use cases or actors, then only the essential use cases should be represented.
5. A use case diagram should describe at least a single module of a system.
6. If the use case diagram is large, then it should be generalized.

An example of a use-case diagram

Following use case diagram represents the working of the student management system:



In the above use case diagram, there are two actors named student and a teacher. There are a total of five use cases that represent the specific functionality of a student management system. Each actor interacts with a particular use case. A student actor can check attendance, timetable as well as test marks on the application or a system. This actor can perform only these interactions with the system even though other use cases are remaining in the system.

It is not necessary that each actor should interact with all the use cases, but it can happen.

The second actor named teacher can interact with all the functionalities or use cases of the system. This actor can also update the attendance of a student and marks of the student. These interactions of both student and a teacher actor together sums up the entire student management application.

When to use a use-case diagram?

A use case is a unique functionality of a system which is accomplished by a user. A purpose of use case diagram is to capture core functionalities of a system and visualize the interactions of various things called as actors with the use case. This is the general use of a use case diagram.

The use case diagrams represent the core parts of a system and the workflow between them. In use case, implementation details are hidden from the external use only the event flow is represented.

With the help of use case diagrams, we can find out pre and post conditions after the interaction with the actor. These conditions can be determined using various test cases.

In general use case diagrams are used for:

1. Analyzing the requirements of a system
2. High-level visual software designing
3. Capturing the functionalities of a system
4. Modeling the basic idea behind the system
5. Forward and reverse engineering of a system using various test cases.

Use cases are intended to convey desired functionality so the exact scope of a use case may vary according to the system and the purpose of creating UML model.

Use case Description:

Sign-up and Sign-in

Name of Use Cases	**Registration**
Description	Once the location has been finalized by the user, the user needs to register in the system to book the tour.
Actors	User
Preconditions	If the user wants to login into the system then they must first sign-up.
Flow	1.User opens the registration page. 2.User fills the details and verify the documents. 3.User will get a confirmation email. 4.User will login into the system.
Alternative flow and Extensions	The system fails to authenticate the user. Then user will be notified and would not be allowed to proceed.
Requirements	User must be connected to internet.

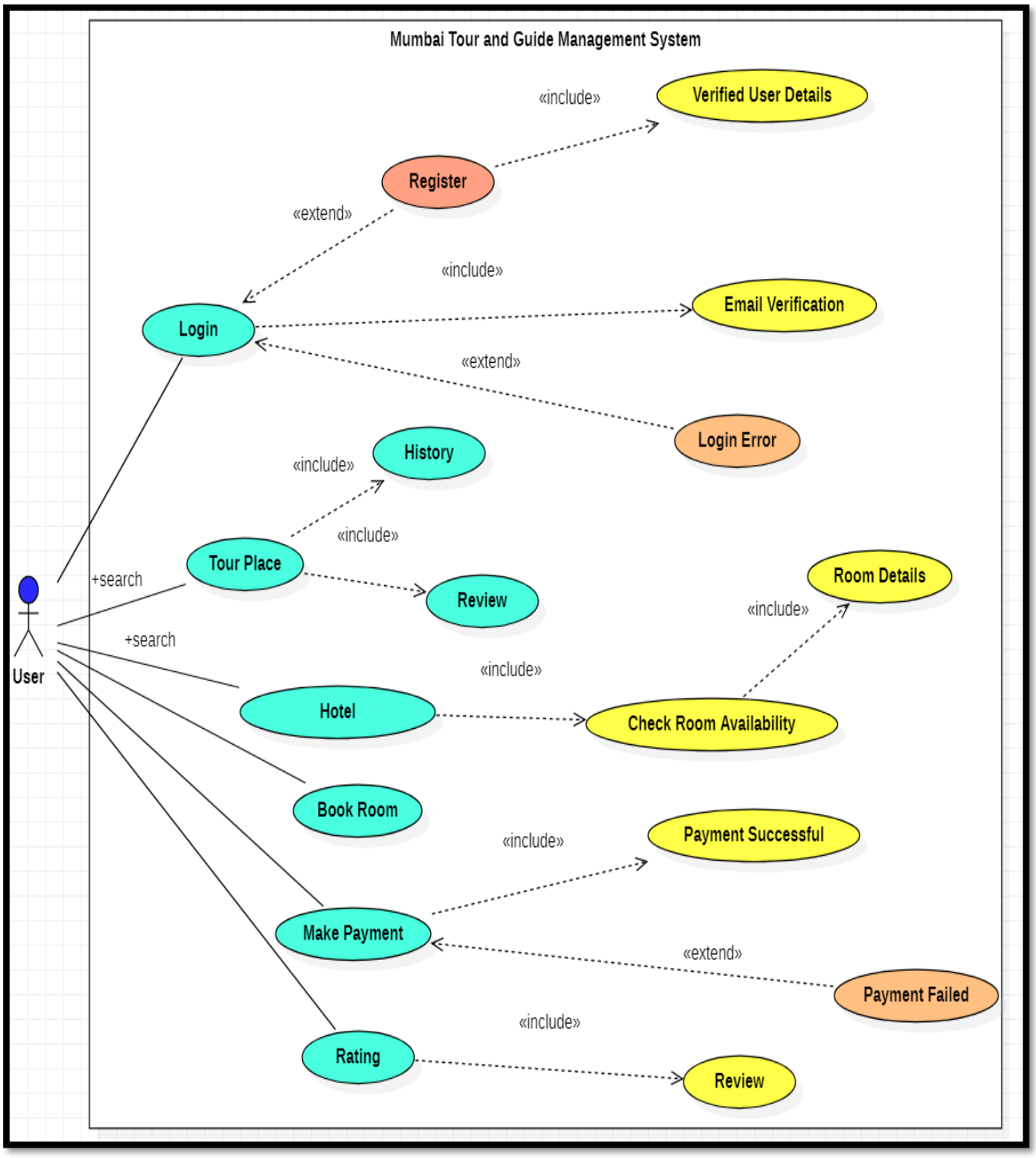
Tour Place:

Name of Use Cases	Tour Place
Description	The user views the different locations and selects the place that he/she wants to visit.
Actors	User
Preconditions	No preconditions are required to view the location.
Flow	1.User visit the website 2.User search for the location. 3.User selects the location based on reviews and images available.
Alternative flow and Extensions	There is no alternative flow available for this process.
Requirements	User must be connected to internet.

Hotel

Name of Use Cases	Hotel
Description	The user will select the hotel based on the reviews and images of the hotel.
Actors	User
Preconditions	The user must be logged-in in the system.
Flow	1.User selects the hotel. 2.User checks the availability of the room. 3.If rooms are available user makes the payment and book the room.
Alternative flow and Extensions	If the rooms are not available then user won't be able to book the room. The user then have to select the another hotel.
Requirements	User must be connected to internet and must be logged-in.

Use case Diagram:



Conclusion:

Thus, use case diagrams are a way to capture the system's functionality and requirements in UML diagrams. It represents a distinct functionality of a system, a component, a package, or a class. We understood the concept of use case diagram as well as its notations. A purpose of use case diagram is to capture the core functionalities of a system.