**EXPERIMENT NO. 4**

**USE CASE DIAGRAM**

Use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

Purpose of Use Case Diagram

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

It gathers the system's needs.

It depicts the external view of the system.

It recognizes the internal as well as external factors that influence the system.

It represents the interaction between the actors.

USE CASE NOTATION:

1. Use Case

UML use case

A use case represents a user goal that can be achieved by accessing the system or software application. In Visual Paradigm, you can make use of the sub-diagram feature to describe the interaction between user and system within a use case by creating a sub-sequence diagram under a use case. You can also describe the use case scenario using the Flow of Events editor.

1. Association



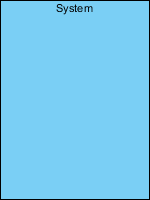
Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association correspond to a sequence of actions between the actor and use case in achieving the use case.

1. Actor

UML actor

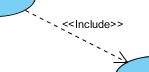
Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So, an actor may be people, computer hardware, other systems, etc.

1. System



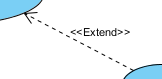
The scope of a system can be represented by a system (shape), or sometimes known as a system boundary. The use cases of the system are placed inside the system shape, while the actor who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.

1. Include



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

1. Extend



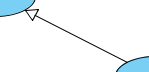
An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.

1. Dependency



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

1. Generalization



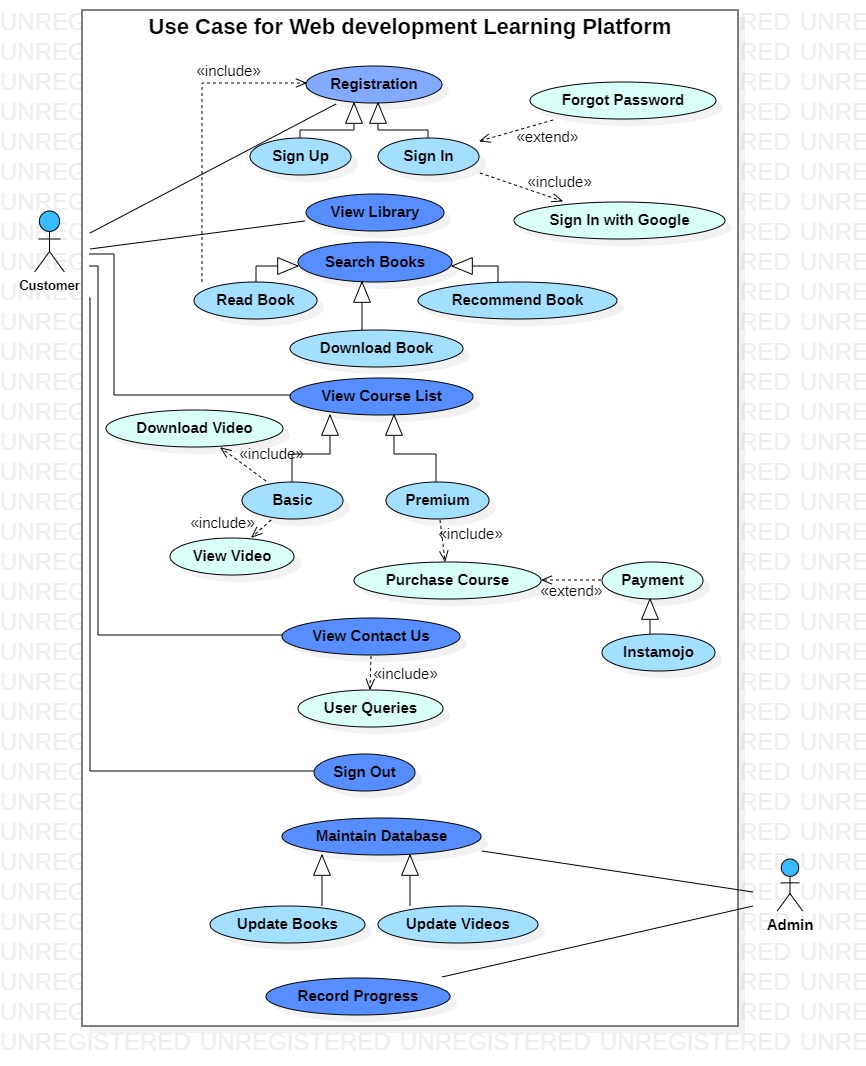
A generalization relationship is used to represent inheritance relationship between model elements of same type. The more specific model element share the same specification with. the more general the model element but carries more details in extra.

1. Realization



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

**USE CASE DIAGRAM FOR WEB DEVELOPMENT LEARNING PLATFORM**



**CLASS DIAGRAM**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

Purpose of Class Diagrams

The purpose of class diagram is to model the static view of an application. Class diagrams are the only diagrams which can be directly mapped with object-oriented languages and thus widely used at the time of construction.

The purpose of the class diagram can be summarized as −

Analysis and design of the static view of an application.

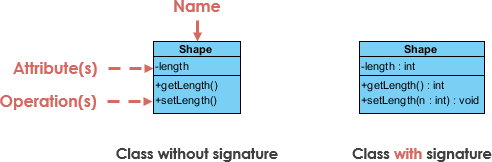
Describe responsibilities of a system.

Base for component and deployment diagrams.

Forward and reverse engineering.

UML Class Notation:

A class represent a concept which encapsulates state (attributes) and behavior (operations). Each attribute has a type. Each operation has a signature. The class name is the only mandatory information.



1. Class Name:

The name of the class appears in the first partition.

1. Class Attributes:

Attributes are shown in the second partition.

The attribute type is shown after the colon.

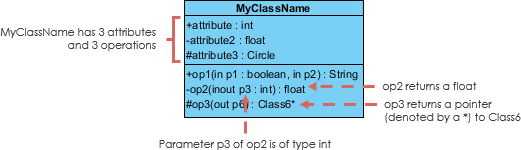
Attributes map onto member variables (data members) in code.

1. Class Operations (Methods):

Operations are shown in the third partition. They are services the class provides.

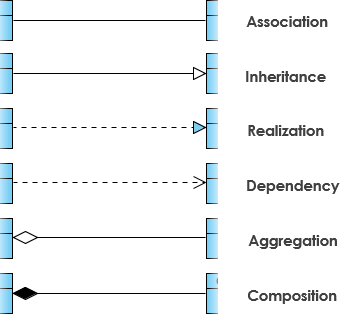
The return type of a method is shown after the colon at the end of the method signature.

The return type of method parameters are shown after the colon following the parameter name. Operations map onto class methods in code



1. Relationships between classes

UML is not just about pretty pictures. If used correctly, UML precisely conveys how code should be implemented from diagrams. A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:



1. Inheritance (or Generalization):

A generalization is a taxonomic relationship between a more general classifier and a more specific classifier. Each instance of the specific classifier is also an indirect instance of the general classifier. Thus, the specific classifier inherits the features of the more general classifier.

1. Association

Associations are relationships between classes in a UML Class Diagram. They are represented by a solid line between classes. Associations are typically named using a verb or verb phrase which reflects the real world problem domain.

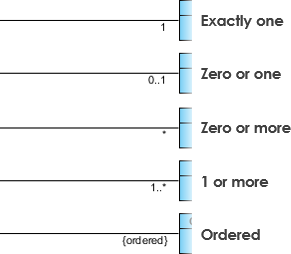
1. Cardinality

Cardinality is expressed in terms of:

one to one

one to many

many to many



1. Aggregation

A special type of association.

It represents a "part of" relationship.

.

1. Composition

A special type of aggregation where parts are destroyed when the whole is

destroyed.

1. Dependency

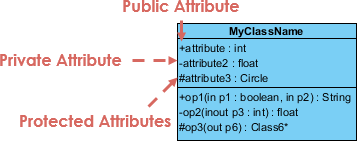
An object of one class might use an object of another class in the code of a method. If the object is not stored in any field, then this is modelled as a dependency relationship.

1. Realization

Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class. In other words, you can understand this as the relationship between the interface and the implementing class.

1. Class Visibility

The +, - and # symbols before an attribute and operation name in a class denote the visibility of the attribute and operation.



+ denotes public attributes or operations

- denotes private attributes or operations

# denotes protected attributes or operations

**CLASS DIAGRAM FOR WEB DEVELOPMENT LEARNING PLATFORM**

