Developing Use cases

In software and systems engineering, a use case is a list of actions or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an *actor*) and a system, to achieve a goal. The actor can be a human, an external system, or time. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. Another way to look at it is a use case describes a way in which a real-world actor interacts with the system. In a system use case you include high-level implementation decisions. System use cases can be written in both an informal manner and a formal manner

Importance:

The advantages of Use cases includes:

- The list of goal names provides the shortest summary of what the system will offer
- It gives an overview of the roles of each and every component in the system. It will help us in defining the role of users, administrators etc.
- It helps us in extensively defining the user's need and exploring it as to how it will work.
- It provides solutions and answers to many questions that might pop up if we start a project unplanned.

Parts of Use Cases

Use Case: What is the main objective of this use case. For eg. Adding a software component, adding certain functionality etc.

Primary Actor: Who will have the access to this use case. In the above examples, administrators will have the access.

Scope: Scope of the use case

Level: At what level the implementation of the use case be.

Flow: What will be the flow of the functionality that needs to be there. More precisely, the work flow of the use case.

Some other things that can be included in the use cases are:

- **Preconditions**
- **Postconditions**
- **Brief course of action**
- **Time Period**

The first step in writing a use case is to define the set of "actors" that will be involved in the story. Actors are the different people (or devices) that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people (or devices) play as the system operates.

It is important to note that an actor and an end user are not necessarily the same thing. A typical user may play a number of different roles when using a system, whereas an actor represents a class of external entities (often, but not always, people) that play just one role in the context of the use case.

Because requirements elicitation is an evolutionary activity, not all actors are identified during the first iteration. It is possible to identify primary actors during the first iteration and secondary actors as more is learned about the system. Primary actors interact to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software. Secondary actors support the system so that primary actors can do their work. Once actors have been identified, use cases can be developed.

Use cases are used to answer following questions:

- Who is the primary actor, the secondary actor(s)?
- What are the actor's goals?
- What preconditions should exist before the story begins?
- What main tasks or functions are performed by the actor?
- What exceptions might be considered as the story is described?
- What variations in the actor's interaction are possible?
- What system information will the actor acquire, produce, or change?
- Will the actor have to inform the system about changes in the external environment?
- What information does the actor desire from the system?
- Does the actor wish to be informed about unexpected changes?

Example for use case: a use case for a software (Home security system)