**Main types of software requirement can be of 3 types:**

* Functional requirements
* Non-functional requirements
* Domain requirements



**Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform. All these functionalities need to be necessarily incorporated into the system as a part of the contract. These are represented or stated in the form of input to be given to the system, the operation performed and the output expected. They are basically the requirements stated by the user which one can see directly in the final product, unlike the non-functional requirements. For example, in a hospital management system, a doctor should be able to retrieve the information of his patients. Each high-level functional requirement may involve several interactions or dialogues between the system and the outside world. In order to accurately describe the functional requirements, all scenarios must be enumerated. There are many ways of expressing functional requirements e.g., natural language, a structured or formatted language with no rigorous syntax and formal specification language with proper syntax. Functional Requirements in Software Engineering are also called Functional Specification.

**Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract.Nonfunctional requirements, not related to the system functionality, rather define how the system should perform The priority or extent to which these factors are implemented varies from one project to other. They are also called non-behavioral requirements. They basically deal with issues like:

* Portability
* Security
* Maintainability
* Reliability
* Scalability
* Performance
* Reusability

**Domain requirements:** Domain requirements are the requirements which are characteristic of a particular category or domain of projects. Domain requirements can be functional or nonfunctional. Domain requirements engineering is a continuous process of proactively defining the requirements for all foreseeable applications to be developed in the software product line. The basic functions that a system of a specific domain must necessarily exhibit come under this category.

**Advantages of classifying software requirements include:**

1. **Better organization:** Classifying software requirements helps organize them into groups that are easier to manage, prioritize, and track throughout the development process.
2. **Improved communication:** Clear classification of requirements makes it easier to communicate them to stakeholders, developers, and other team members. It also ensures that everyone is on the same page about what is required.
3. **Increased quality:** By classifying requirements, potential conflicts or gaps can be identified early in the development process. This reduces the risk of errors, omissions, or misunderstandings, leading to higher quality software.
4. **Improved traceability:** Classifying requirements helps establish traceability, which is essential for demonstrating compliance with regulatory or quality standards.

**Disadvantages of classifying software requirements include:**

1. **Complexity**: Classifying software requirements can be complex, especially if there are many stakeholders with different needs or requirements. It can also be time-consuming to identify and classify all the requirements.
2. **Rigid structure**: A rigid classification structure may limit the ability to accommodate changes or evolving needs during the development process. It can also lead to a siloed approach that prevents the integration of new ideas or insights.
3. **Misclassification**: Misclassifying requirements can lead to errors or misunderstandings that can be costly to correct later in the development process.