**Functional vs Non Functional Requirements**

* Difficulty Level : [Basic](https://www.geeksforgeeks.org/basic/)
* Last Updated : 29 Apr, 2020

Requirements analysis is very critical process that enables the success of a system or software project to be assessed. Requirements are generally split into two types: *Functional* and *Non-functional requirements*.

**Functional Requirements:** These are the requirements that the end user specifically demands as basic facilities that the system should offer. 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.

**Non-functional requirements:** These are basically the quality constraints that the system must satisfy according to the project contract. 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
* Flexibility

Following are the differences between Functional and Non Functional Requirements

| **Functional Requirements** | **Non Functional Requirements** |
| --- | --- |
| A functional requirement defines a system or its component. | A non-functional requirement defines the quality attribute of a software system. |
| It specifies “What should the software system do?” | It places constraints on “How should the software system fulfill the functional requirements?” |
| Functional requirement is specified by User. | Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers. |
| It is mandatory. | It is not mandatory. |
| It is captured in use case. | It is captured as a quality attribute. |
| Defined at a component level. | Applied to a system as a whole. |
| Helps you verify the functionality of the software. | Helps you to verify the performance of the software. |
| Functional Testing like System, Integration, End to End, API testing, etc are done. | Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done. |
| Usually easy to define. | Usually more difficult to define. |
| **Example**  **1)** Authentication of user whenever he/she logs into the system. **2)** System shutdown in case of a cyber attack. **3)** A Verification email is sent to user whenever he/she registers for the first time on some software system. | **Example**  **1)** Emails should be sent with a latency of no greater than 12 hours from such an activity. **2)** The processing of each request should be done within 10 seconds **3)** The site should load in 3 seconds when the number of simultaneous users are > 10000 |

**Non-functional Requirements in Software Engineering**

* Last Updated : 04 Jun, 2020

**Non-Functional Requirements** are the constraints or the requirements imposed on the system. They specify the quality attribute of the software. Non-Functional Requirements deal with issues like scalability, maintainability, performance, portability, security, reliability, and many more. Non-Functional Requirements address vital issues of quality for software systems. If NFRs not addressed properly, the results can include:

* Users, clients, and developers are unsatisfied.
* Inconsistent software.
* Time and cost overrun to fix the software which was prepared without keeping NFRs in mind.

Types of Non-functional Requirement :

1. Scalability
2. Reliability
3. Regulatory
4. Maintainability
5. Serviceability
6. Utility
7. Security
8. Manageability
9. Data integrity
10. Capacity
11. Regulatory
12. Availability
13. Usability
14. Interoperability
15. Environmental

These can can be classified as :

1. **Performance constraints –**  
   Reliability, security, response time, etc.
2. **Operating constraints –**  
   These include physical constraints (size, weight), personnel availability, skill level considerations, system accessibility for maintenance, etc.
3. **Interface constraints –**  
   These describe how the system is to interface with its environment, users, and other systems. For example, user interfaces and their qualities (e.g., user-friendliness).
4. **Economic constraints –**  
   Immediate and/or long-term costs.
5. **Lifecycle requirements –** Quality of the design:  
   These measured in terms such as maintainability, enhance ability, portability.

**Advantages of Non-Functional Requirement :**

* They ensure the software system follows legal and adherence rules.
* They specify the quality attribute of the software.
* They ensure the reliability, availability, performance, and scalability of the software system
* They help in constructing the security policy of the software system.
* They ensure good user experience, ease of operating the software, and minimize the cost factor.

**Disadvantages of Non-functional requirement :**

* The nonfunctional requirement may affect the various high-level software subsystem.
* They generally increase the cost as they require special consideration during the software architecture/high-level design phase.
* It is difficult to change or alter non-functional requirements once you pass them to the architecture phase.