SOFTWARE ENGINEERING

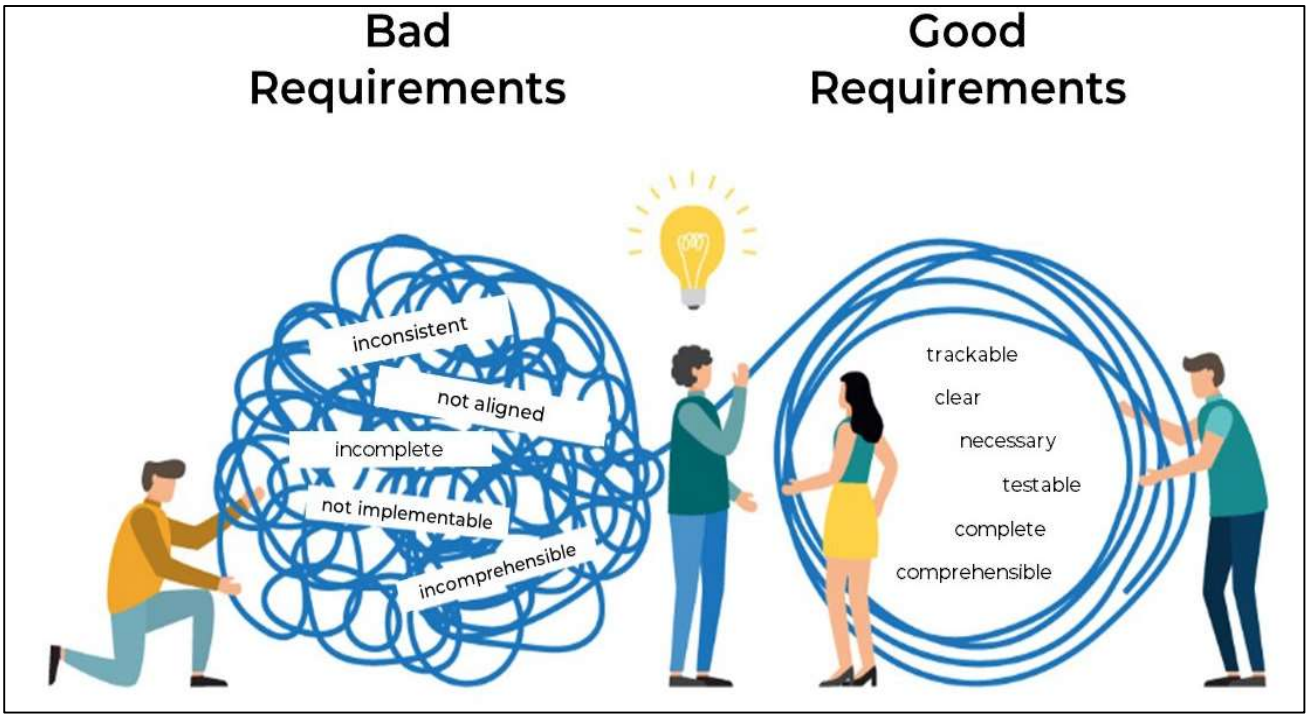
UNIT – 2

TOPIC – 1

UNDERSTANDING REQUIREMENTS

Software requirements describe what a software system should do and how it should perform. These requirements guide developers in building the system the way users expect. Requirements are usually divided into two main types:

1. Functional requirements and
2. Non-functional requirements.



1. Functional Requirements

Functional requirements describe the specific actions the system must perform. They outline the functions or services that the software provides to users. Essentially, they define what the system should do.

Examples of Functional Requirements:

i. A system must allow users to search for items in a database.
ii. Every order made in the system should generate a unique order ID.
iii. The system should let users view documents.

Why Functional Requirements Are Important:

These requirements make sure the software does the tasks it's supposed to do. They cover what happens when a user gives input (like clicking a button) and what the system should output (like showing search results).

Key Qualities of Good Functional Requirements:

i. **Complete:** All required services should be included.
ii. **Consistent:** There should be no contradictions between different requirements.

2. Non-Functional Requirements

Non-functional requirements describe how well the system should perform, not what it should do. These are qualities or attributes that the system should have, such as how fast it should be, how secure, or how reliable.

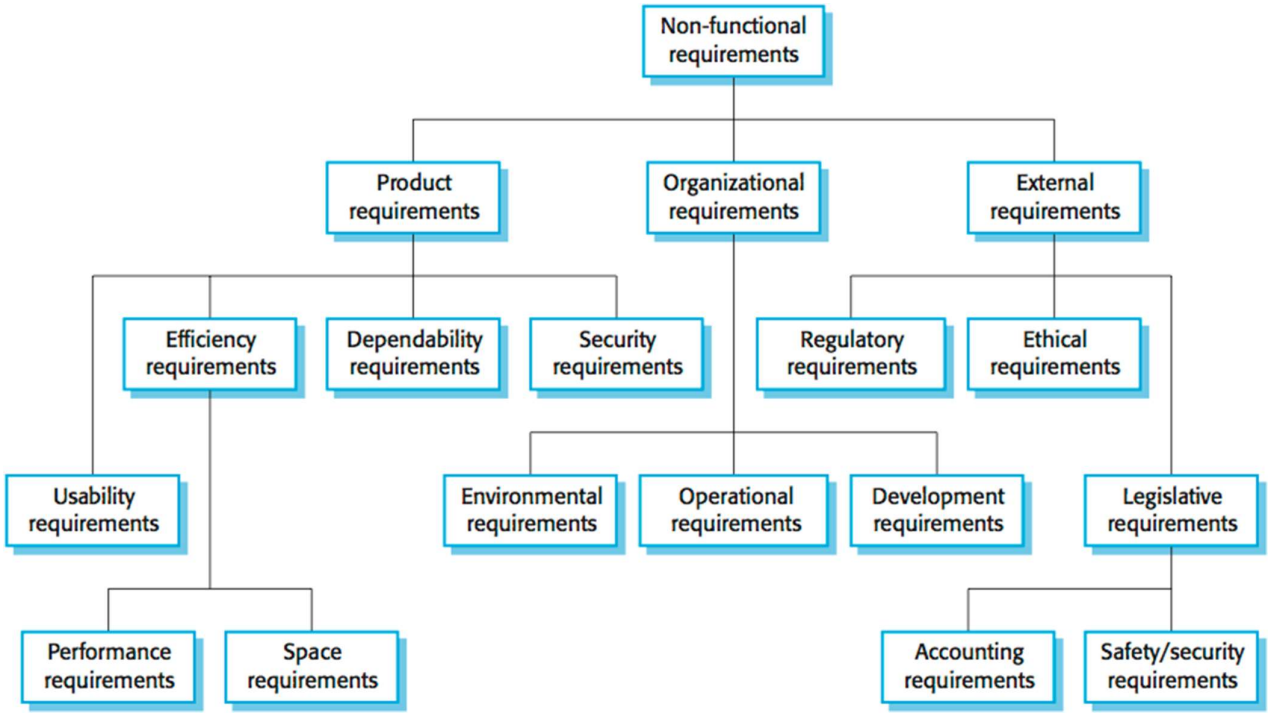
Examples of Non-Functional Requirements:

i. The system must respond to user actions within 2 seconds.
ii. The system should be able to handle up to 1,000 users at the same time.
iii. The system should be available 99% of the time (high reliability).

Why Non-Functional Requirements Are Important:

These requirements ensure that the software meets user expectations beyond just functionality. If the system is too slow or unreliable, it won't satisfy users, even if it performs the right tasks.

Non – Functional Requirements Types:



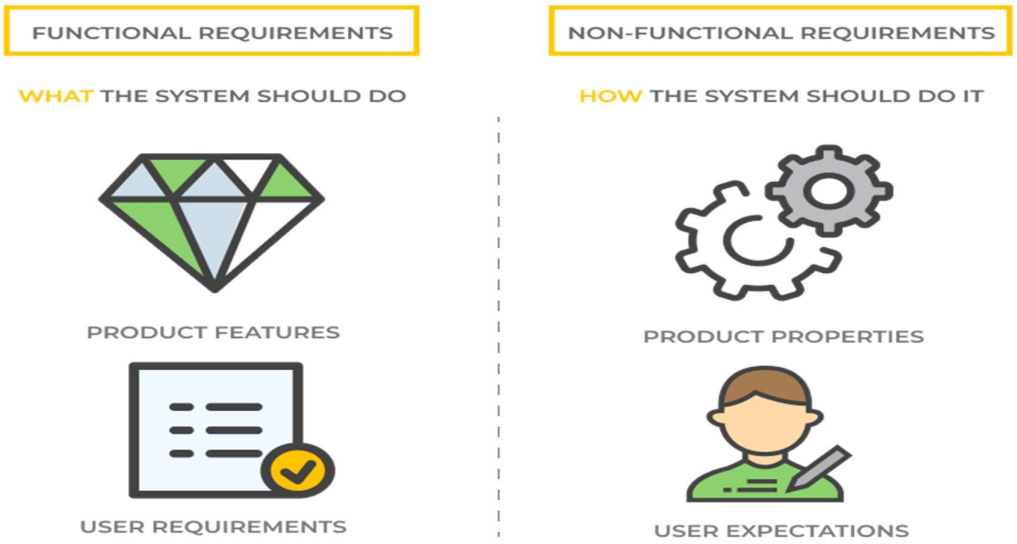
a. **Product Requirements:** These are all about how the software should perform and behave. They focus on what you want the software to do in terms of speed, efficiency, and other technical details. Think of them as the "performance expectations" for your software.
   Speed: How fast should it be? For example, it should find your files in under 2 seconds.
   Memory Usage: How much memory should it use? It shouldn't use more than 500MB of your computer's memory.
   Example: The software must use less than 500MB of memory. This means it should not take up more than 500MB of your computer's memory while running.
b. **Organizational Requirements**: These are related to the company's own rules or policies. They dictate the choices you make when developing software, based on what the company prefers or requires.
   Programming Language: The company might require using a specific language to build the software. For example, they might want it built with Python.
   Example: The software must be built using Python. This means the development team has to use Python for creating the software.

c. **External Requirements:** These come from outside the software itself and usually involve external factors that the software must comply with or interact with.

Laws and Regulations: The software needs to comply with legal rules, like protecting user privacy.

Other Systems: The software might need to work with other systems or services.

Example: The system must follow data privacy laws. This means it has to handle personal information in a way that meets legal privacy standards.



Difference between the functional and non-functional requirements.

Aspect	Functional Requirement	Non-Functional Requirement
What it Describes	Actions or tasks the system should perform	Qualities of the system (speed, security)
Requirement Type	Mandatory	Often not mandatory, but still important
Captured in	Use cases (scenarios of how users interact)	Quality attributes (e.g., performance metrics)
Focus	What the system does	How well the system performs
Testing	Functional testing (e.g., feature testing)	Non-functional testing (e.g., performance tests)

Aspect	Functional Requirement	Non-Functional Requirement
Example	"The system must allow users to place orders."	"The system must handle 1,000 orders per second."

User Requirements

User Requirements are about what users want from the software, written in plain language that's easy to understand. They focus on what the software should do from the user's point of view.

- **Functional User Requirements:** These describe what the software should do. For example, if a user wants to easily manage customer orders, that's what the software needs to be able to do.

Example: "I want the software to allow me to manage my customer orders easily." This means the software should help users handle their orders without any trouble.

- **Non-Functional User Requirements:** These describe how the software should perform or feel. It's about the user's experience, like how fast or easy it is to use.

Example: "The system should be fast and easy to use." This means users want the software to respond quickly and be simple to navigate.

User Requirements Are Gathered by:

- **Interviews:** Talking directly with users to find out what they need.
- **Surveys:** Sending questions to users to get their opinions and needs.
- **Workshops:** Bringing users and developers together to discuss what the software should do.

System Requirements

System Requirements are technical details about how the software should work. They translate user needs into specific instructions for developers to follow when building the software.

- **Functional System Requirements:** These describe what the system should do in technical terms. For example, how the software should handle customer orders.

Example: "The system shall store customer orders in a database and generate unique order IDs." This means the software should save orders in a database and create a unique ID for each one.

- **Non-Functional System Requirements:** These describe how the system should perform in technical terms, like how many users it can handle at once or how quickly it should respond.

Example: "The system shall support up to 500 users at the same time without slowing down." This means the software should be able to handle 500 users using it at the same time without becoming slow.

Purpose of System Requirements:

- **Guide Development:** They tell developers exactly what to build and how to build it.
- **Testing:** They help check if the finished software meets the user's needs.

Difference between the user and system requirements.

Aspect	User Requirements	System Requirements
Purpose	Capture what users want from the system	Define how the system will technically fulfill user needs
Language	Simple, non-technical	Technical, for developers and engineers
Who Uses Them	Users, stakeholders, business analysts	Developers, testers, architects
Detail Level	High-level, broad goals	Detailed, specific instructions
Examples	"The system should allow me to view my orders."	"The system shall retrieve order data from the database and display it."