Requirements

What are Requirements?

Requirements are detailed descriptions of the conditions, capabilities, or constraints needed to achieve a specific objective within a project or system. They define what a product, service, or system must do to fulfil the expectations of stakeholders, including businesses, users, and technical teams. Requirements act as a foundation for project planning, development, and validation, ensuring all parties involved share a clear understanding of the desired outcomes. They bridge the gap between business needs and technical implementation, guiding the team toward building a solution that aligns with organizational goals and user expectations. Well-defined requirements are crucial for managing project scope, minimizing misunderstandings, and delivering successful results.

Why do Requirements matter?

- Requirements provide a detailed understanding of what the project is expected to achieve, outlining its scope and boundaries. This clarity ensures that all stakeholders are aligned on the project's goals and reduces the risk of scope creep.
- They act as a common language between stakeholders, such as business teams, end users, and technical teams. This shared understanding minimizes misunderstandings and fosters effective collaboration across different roles.
- Requirements serve as a blueprint for the technical team, guiding the design, development, and testing processes. They ensure that the solution is built to meet the specific needs and objectives identified at the outset.
- By clearly identifying needs and objectives, requirements help teams prioritize features or tasks based on their importance, value, or urgency. This is especially critical in projects with limited time or resources.
- Requirements form the basis for testing and quality assurance, helping to validate that the solution meets stakeholder expectations and verify that it functions as intended. This reduces the risk of delivering a product that fails to satisfy users.
- Clear requirements reduce uncertainties by addressing potential risks and ambiguities early in the project lifecycle. They help prevent costly changes or rework later in the development process.

- Requirements focus on delivering value to stakeholders by explicitly addressing their needs and expectations. This leads to higher levels of satisfaction and greater acceptance of the final product.
- With well-documented requirements, teams can trace each development activity back to specific business needs. This ensures accountability and helps teams track progress while demonstrating compliance with project goals.
- By defining what is necessary upfront, requirements help in creating realistic timelines and budgets. They minimize wasted effort on unnecessary features or misaligned solutions, saving time and money.
- Requirements provide a baseline that can be reviewed and updated as the project evolves. This iterative process allows teams to adapt to changes while maintaining alignment with the project's overall objectives.

Characteristics of Good Requirements

1. **SMART**

   The SMART framework ensures that requirements are effective and actionable by meeting the following criteria:

   - **Specific –** Requirements must be precise and unambiguous. They should clearly define what is needed, avoiding vague or generalized statements.
     Example : Instead of saying, "The system should provide reports," specify, "The system must generate daily sales reports with data segmented by region."

   - **Measurable -** Requirements should include criteria to measure their fulfillment. This ensures that stakeholders can verify whether the requirement has been met.
     Example : "The system must support 100 concurrent users without performance degradation."

   - **Achievable -** Requirements must be realistic and feasible within the given constraints, such as time, budget, and resources.
     Example : Avoid requiring a small team to develop a highly complex system in an unrealistic timeframe.

   - **Relevant –** Every requirement must align with the business goals and the project's overall objectives. Irrelevant requirements add unnecessary complexity.
     Example : A feature like social media integration may not be relevant for an internal employee management system.

   - **Time Bound -** Each requirement should include a deadline or timeframe for implementation to guide project planning.
     Example : "The payment gateway must be integrated by the end of the second development sprint."

2. **CLEAR**

   The CLEAR framework focuses on clarity and simplicity, ensuring that requirements are easy to understand and implement:

- **Collaborative –** Requirements should be developed with input from all relevant stakeholders, including business teams, end users, and technical teams. This ensures they address all perspectives and needs.

- **Limited –** Requirements should focus on essential features or functionality, avoiding overloading the project with unnecessary details.
  Example : Concentrate on key features for the Minimum Viable Product (MVP) rather than attempting to build a fully-fledged solution immediately.

- **Essential –** Requirements must address core needs critical to the project's success. Secondary features can be deferred to later stages.
  Example : Prioritize developing a secure login system before adding optional features like theme customization.

- **Achievable –** Similar to SMART, CLEAR emphasizes feasibility, ensuring that requirements are realistic within the project's constraints.

- **Refinable –** Requirements should be flexible to accommodate changes as the project evolves. This is particularly crucial in Agile methodologies.
  Example : Initial requirements for a reporting dashboard may be refined based on user feedback after a prototype is reviewed.

3. **INVEST**

   Primarily used in Agile environments for writing user stories, the INVEST framework ensures that requirements are well-structured and manageable:

   - **Independent –** Each requirement or user story should be self-contained and not rely on others. This independence allows for parallel development and testing.
     Example : "As a user, I want to reset my password via email" can be developed independently of other user management features.

   - **Negotiable –** Requirements should be open to discussion and refinement. This flexibility ensures that stakeholders can collaborate to arrive at the best solution.
     Example : The design of a feature can be adjusted based on feasibility discussions with the technical team.

- **Valuable –** Each requirement must deliver clear value to the stakeholders or end users. If it doesn't contribute to the project's goals, it should be reconsidered.
  Example : "As a customer, I want to view my order history" adds value by improving user convenience.

- **Estimable –** Requirements should be described in enough detail to allow the team to estimate the effort, time, and resources needed for implementation.
  Example : Clearly state, "The system must support exporting reports in PDF and Excel formats."

- **Small –** Requirements should be broken down into manageable pieces to ensure they can be completed within a single sprint or iteration.
  Example : Instead of "Create a reporting system," specify, "Develop a monthly sales report feature."

- **Testable –** Requirements should include criteria that can be validated through testing to confirm they are met.
  Example : "The system must send an email notification within 30 seconds of order placement."

4. **6 Cs of Requirements**

   This principle emphasizes six key attributes that make requirements clear, concise, and actionable:

   - **Clear –** Requirements must be unambiguous and easy to understand. This ensures all stakeholders interpret them in the same way.
     Example : Avoid phrases like "The system should perform efficiently" and instead specify, "The system should load pages within 2 seconds."

   - **Concise –** Requirements should be brief yet comprehensive, avoiding unnecessary details that could cause confusion.
     Example : "The system must generate invoices" is better than a lengthy paragraph describing the invoice's layout.

- **Consistent –** Requirements should not conflict with one another or with project goals. Consistency ensures coherence across all project documentation.
  Example : If one requirement states “The system must support Internet Explorer,” another requirement should not exclude it.

- **Complete –** Requirements should provide all the necessary information to avoid gaps or assumptions.
  Example : Instead of “Provide reports,” specify the types of reports, data sources, and formats required.

- **Correct –** Requirements must accurately reflect stakeholder needs and the project’s objectives. Regular reviews ensure correctness.
  Example : Confirm with stakeholders that the requirement “The system must integrate with PayPal” aligns with their payment processing needs.

- **Controlled –** Requirements should be managed and tracked for changes to maintain project integrity. Use tools like Jira or Trello for version control.
  Example : Maintain a change log to document updates to a requirement.

Hierarchy of Requirements

The hierarchy of requirements organizes requirements into levels to ensure they align with business objectives, meet user needs, and translate effectively into technical solutions. This structure helps streamline the development process by connecting high-level goals with actionable details.

1. Business Requirements

Business requirements define the high-level goals and objectives an organization aims to achieve through a project. These requirements articulate the why of the initiative, focusing on the value it will deliver to the business and its stakeholders. They provide the foundation for all subsequent requirements by aligning the project with strategic goals, addressing problems, or leveraging opportunities. Business requirements are broad and non-technical, ensuring stakeholders across the organization understand the intended outcomes. These requirements are typically documented in a Business Requirements Document (BRD), which serves as a guide for decision-making and scope management.

Example :

- Increase online sales by 20% within the next year.
- Enhance customer satisfaction scores by 15% through faster service delivery.
- Reduce operational costs by automating manual processes.
- Expand the market share in a specific geographic region by launching a new product line.

2. User Requirements

User requirements define the specific needs and expectations of the end users from the system. They describe what the system should do to enable users to achieve their goals and tasks effectively. These requirements focus on user interactions, usability, and the desired functionality of the system. User requirements are typically documented in a Functional Requirements Document (FRD) and are expressed in non-technical terms, making them accessible to both technical teams and stakeholders. These requirements often take the form of use cases, user stories, or scenarios, focusing on user goals and system behavior.

Example :

- Customers must be able to view and search for products by category.

- Users should receive a confirmation email after placing an order.
- Customers must have the ability to reset their passwords securely.
- Administrators need access to an analytics dashboard for tracking sales metrics.

3. System Requirements

System requirements specify the how of the system, detailing the technical and functional specifications needed to implement the user requirements. These requirements are precise, measurable, and technical, providing developers, engineers, and testers with the information required to design, build, and maintain the system. System requirements are documented in a Software Requirements Specification (SRS), which covers both functional and non-functional requirements. Functional requirements outline the system's capabilities, while non-functional requirements address performance, security, and other quality attributes.

Example :

- Functional: The system must validate user credentials during login.
- Non-functional: The system should process up to 500 transactions per minute.
- Security: The system must encrypt sensitive data using 256-bit SSL encryption.