Requirements In a system

Requirements in a system are the specific needs or conditions that the system must fulfill. These can be functional, describing specific actions the system must perform, or non-functional, relating to performance or security standards.

Types of Requirements:

1. **Functional Requirements:** Define what the system should do.
   - *Example:* The system must allow users to log in and access their accounts.
2. **Non-Functional Requirements:** Define how the system should perform.
   - *Example:* The system must load pages within 2 seconds.
3. **Business Requirements:** High-level needs of the organization.
   - *Example:* The system should support 500 concurrent users.
4. **User Requirements:** Specific needs of the end users.
   - *Example:* The user interface should be intuitive and easy to navigate.

Importance of Requirements:

1. **Clarity:** Provides a clear understanding of what is needed, reducing ambiguity.
2. **Guidance:** Helps in planning and designing the system effectively.
3. **Quality Assurance:** Sets standards for testing to ensure the system meets expectations.
4. **Resource Management:** Assists in allocating resources efficiently and avoiding unnecessary costs.
5. **Stakeholder Alignment:** Ensures all parties have a common understanding, reducing the risk of misunderstandings.

Steps to Gather Requirements:

1. **Identify Stakeholders:**
   - Determine who will use or be affected by the system.
   - *Example:* Users, project managers, IT staff.
2. **Conduct Interviews:**
   - Speak directly with stakeholders to understand their needs and expectations.
   - *Example:* Interviewing users about the features they want in a new application.
3. **Use Questionnaires:**
   - Distribute surveys to collect information from a larger audience.
   - *Example:* Sending out surveys to gather feedback on desired features.
4. **Hold Workshops:**
   - Facilitate group sessions to brainstorm and discuss requirements.
   - *Example:* Organizing workshops where users and developers discuss system functionalities.
5. **Observe Users:**
   - Study how users interact with the current system to identify areas for improvement.
   - *Example:* Observing users struggling with a complicated login process.
6. **Analyze Documents:**
   - Review existing documentation to understand current processes and requirements.
   - *Example:* Reading through user manuals and previous system documentation.
7. **Create Use Cases:**

    - Develop detailed scenarios that describe how users will interact with the system.
    - *Example:* Writing use cases that outline the steps a user takes to log in and perform tasks.
8. **Prototype:**
    - Develop a simple model of the system to gather feedback and refine requirements.
    - *Example:* Creating a basic version of the new app to demonstrate and get user input.
9. **Validate Requirements:**
    - Confirm that the gathered requirements are accurate and complete by reviewing them with stakeholders.
    - *Example:* Presenting the requirements document to stakeholders for approval and making necessary adjustments.

Design requirements traceability is crucial to ensure that every system requirement is accounted for in the design and implementation phases. Here's an overview of a typical requirements traceability path:

1. Requirements Identification:

- Each requirement is assigned a unique identifier.
- *Example:* Functional requirement FR-001 for user login functionality.

2. Traceability Matrix:

- A traceability matrix maps requirements to design components, ensuring each requirement is addressed.
- *Example:* FR-001 maps to design component DC-01 (Login Module).

3. Design Documentation:

- Detailed design documents describe how each requirement will be implemented.
- *Example:* The Login Module design document outlines the architecture, data flow, and user interface.

4. Development:

- Developers use the design documents to build the system components.
- *Example:* Developers implement the Login Module as per the design specifications.

5. Testing:

- Each requirement is tested to ensure it has been implemented correctly.
- *Example:* Test case TC-001 verifies that FR-001 (user login) works as intended.

6. Validation and Verification:

- The system is validated and verified to confirm that it meets all requirements.
- *Example:* Conducting user acceptance testing (UAT) to ensure the Login Module meets user expectations.

7. Traceability Reports:

- Traceability reports provide an overview of the relationships between requirements, design, development, and testing.
- *Example:* A report showing that FR-001 is linked to DC-01, implemented in the Login Module, and validated by TC-001.