1. **Functional Requirements**:
    - Describe what the system or product should do.
    - Specify features, behaviors, and interactions.
    - Example: "The system must allow users to reset their password via email."
2. **Non-Functional Requirements**:
    - Define how the system should perform or operate.
    - Include attributes like scalability, reliability, security, and performance.
    - Example: "The system must support 1,000 concurrent users with a response time of less than 2 seconds."
3. **System Constraints**:
    - Outline limitations or restrictions on the system.
    - May include hardware, software, or regulatory constraints.
    - Example: "The application must be compatible with Windows 10 and macOS 12."
4. **Interface Requirements**:
    - Specify how the system interacts with users, other systems, or external components.
    - Example: "The API must support RESTful endpoints for data exchange."
5. **Performance Requirements**:
    - Define the expected performance metrics, such as speed, capacity, and efficiency.
    - Example: "The database must process 10,000 transactions per second."
6. **Security Requirements**:
    - Outline measures to protect the system from unauthorized access, data breaches, or other threats.
    - Example: "All user data must be encrypted using AES-256 encryption."
7. **Usability Requirements**:
    - Define the ease of use and user experience expectations.
    - Example: "The interface must be intuitive, with a maximum of three clicks to complete any task."
8. **Compliance Requirements**:
    - Specify adherence to legal, regulatory, or industry standards.
    - Example: "The system must comply with GDPR for data privacy."
9. **Maintainability Requirements**:
    - Define how easily the system can be updated, modified, or repaired.
    - Example: "The codebase must include comprehensive documentation for future maintenance."
10. **Scalability Requirements**:

    - Specify the system's ability to handle growth in users, data, or transactions.
    - Example: "The system must scale to support 10x the current user base without performance degradation."

Importance of Technical Requirements:

- **Clarity**: Provides a clear understanding of what needs to be built.
- **Alignment**: Ensures all stakeholders agree on the system's goals and functionality.
- **Guidance**: Serves as a reference for developers, testers, and project managers.
- **Quality Assurance**: Helps in verifying that the final product meets the specified criteria.
- **Risk Mitigation**: Identifies potential challenges early in the development process.

Documentation Format:

Technical requirements are often documented in a **Software Requirements Specification (SRS)** or a **Technical Requirements Document (TRD)**. These documents include sections for each type of requirement, along with diagrams, use cases, and acceptance criteria.

By defining technical requirements clearly and comprehensively, teams can reduce ambiguity, avoid scope creep, and deliver a product that meets user and business needs.