Non-Functional Requirements

Non-functional requirements, or NFRs, for software define the benchmarks by which a system is assessed for overall performance rather than specific behaviors. They are sometimes called "quality attributes" and typically describe a system's functionality rather than its intended functions. The following is a list of common non-functional requirements.

Performance

* The efficiency and speed at which a system operates are measured by its performance.
* Speed and Response Time: The system should respond fast when a user initiates an activity, such as completing a transaction or loading a page.
* Throughput: It measures how much work the system can do in a given length of time, making it possible to manage multiple requests or tasks at once efficiently.

Reliability

* Reliability is the system's capacity to function steadily and consistently.
* Uptime and Fault Tolerance: The system should always operate without a hitch, reducing downtime even in the event of unanticipated issues.
* Recoverability: In the event of a malfunction, the system should be able to quickly recover and carry on with normal operations with minimizing disruption or data loss.

Security

* The primary objective of security is to keep the system and its data safe from attacks.
* Access Control and Authentication: To ensure that only authorized users can access critical areas, the system needs to employ strong authentication mechanisms.
* Data Integrity and Protection: Data should use encryption and other security measures to ensure that it is protected from unauthorized access and does not alter while being sent or kept.

Capacity

* The capacity of a system indicates how much data and user load it can handle:
* Data Processing and Storage: The system needs to be efficient in order to manage and store enormous volumes of data without sacrificing performance.
* Concurrent User Support: It should be able to handle multiple users or processes at once without stuttering, ensuring that operations continue even under high demand.

Maintainability

* A system is said to be maintainable if it is simple to update and manage.
* Update Ease: A maintainable system permits a small amount of disruption during quick upgrades, new additions, and problem fixes.
* Documentation and Readability: Developers can more easily understand and modify the system when there is clear documentation and well-organized code.

Efficiency

* Efficiency is a system's ability to make the best use of its resources, particularly when it is working under heavy load.
* Resource Usage: To maintain performance, an efficient system uses less memory, CPU, and bandwidth.
* Performance Under Load: When the system is handling heavier loads, there shouldn't be any noticeable decrease in its responsiveness or speed.