**What are Software Requirements Specifications?**

An SRS is basically an organization’s understanding (in writing) of a customer or potential client’s system requirements and dependencies *at a particular point in time* (usually) prior to any actual design or development work. It’s a two-way insurance policy that assures that both the client and the organization understand the other’s requirements from that perspective at a given point in time.

The SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an eCommerce Web site, and so on) must provide, as well as states any required constraints by which the system must abide. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the “parent” document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

It’s important to note that an SRS contains functional and nonfunctional requirements only; it doesn’t offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer’s system requirements to be.

A well-designed, well-written SRS accomplishes four major goals:

* **It provides feedback to the customer.** An SRS is the customer’s assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decision tables, and so on.
* **It decomposes the problem into component parts.** The simple act of writing down software requirements in a well-designed format organizes information, places borders around the problem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
* **It serves as an input to the design specification.** As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised. It serves as a product validation check. The SRS also
* **It serves as the parent document** for testing and validation strategies that will be applied to the requirements for verification.

Software requirements specifications are typically developed during the first stages of “Requirements Development,” which is the initial product development phase in which information is gathered about what requirements are needed–and not. This information-gathering stage can include onsite visits, questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client’s current business environment. The actual specification, then, is written after the requirements have been gathered and analyzed.

<https://techwhirl.com/writing-software-requirements-specifications/>

f there is any one thing any project must have in order not to be doomed to failure, that is a sensible and comprehensive collection of both the functional and non-functional requirements.

**Any project’s requirements need to be well thought out, balanced and clearly understood** by all involved, but perhaps of most importance is that they are **not dropped or compromised halfway through** the project.

[Try ReQtest Free - #1 Software Testing Tool](https://reqtest.com/try-reqtest/)

[Do agile software testing with ReQtest – Try for free](https://reqtest.com/get-full-overview-agile-test-process/)

However, what exactly is the difference between ‘functional’ and ‘non functional’ requirements? It’s not that complex, and once you understand the difference, the definition will be clear.

The official definition of ‘a functional requirement’ is that it essentially **specifies something the system should do.**

Typically, functional requirements will specify a behaviour or function, for example:  
“Display the name, total size, available space and format of a flash drive connected to the USB port.” Other examples are “add customer” and “print invoice”.

A functional requirement for a milk carton would be “ability to contain fluid without leaking”

Some of the more typical functional requirements include:

* Business Rules
* Transaction corrections, adjustments and cancellations
* Administrative functions
* Authentication
* Authorization levels
* Audit Tracking
* External Interfaces
* Certification Requirements
* Reporting Requirements
* Historical Data
* Legal or Regulatory Requirements

So what about Non-Functional Requirements? What are those, and how are they different?

Simply put, the difference is that **non-functional requirements describe how the system works**, while **functional requirements describe what the system should do**.

The definition for a non-functional requirement is that it essentially specifies **how the system should behave** and that it is a constraint upon the systems behaviour. One could also think of non-functional requirements as quality attributes for of a system.

[Try ReQtest Free - #1 Bug Tracking Tool](https://reqtest.com/try-reqtest/)

A non-functional requirement for a hard hat might be “must not break under pressure of less than 10,000 PSI”

Non-functional requirements cover all the remaining requirements which are not covered by the functional requirements. They specify criteria that judge the operation of a system, rather than specific behaviours, for example: “Modified data in a database should be updated for all users accessing it within 2 seconds.”

Some typical non-functional requirements are:

* Performance – for example Response Time, Throughput, Utilization, Static Volumetric
* Scalability
* Capacity
* Availability
* Reliability
* Recoverability
* Maintainability
* Serviceability
* Security
* Regulatory
* Manageability
* Environmental
* Data Integrity
* Usability
* Interoperability

As said above, non-functional requirements specify the system’s ‘quality characteristics’ or ‘quality attributes’.

Many different stakeholders have a vested interest in getting the non-functional requirements right particularly in the case of large systems where the buyer of the system is not necessarily also the user of the system.

The importance of non-functional requirements is therefore not to be trifled with. One way of ensuring that as few as possible non-functional requirements are left out is to use non-functional requirement groups. For an explanation on how to use non-functional requirement group, [read this blog post](http://www.reqtest.com/newsletters/what-are-non-functional-requirements/) which will give you four of the main groups to use.