A **software requirements specification** (**SRS**) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

The below diagram depicts the various types of requirements that are captured during SRS.



Goals[[edit](https://en.wikipedia.org/w/index.php?title=Software_requirements_specification&action=edit&section=2" \o "Edit section: Goals)]

The Software Requirements Specification (SRS) is a communication tool between stakeholders and software designers. The specific goals of the SRS are:

* Facilitating reviews
* Describing the scope of work
* Providing a reference to software designers (i.e. navigation aids, document structure)
* Providing a framework for testing primary and secondary use cases
* Including [features](https://en.wikipedia.org/wiki/Software_feature) to customer requirements
* Providing a platform for ongoing refinement (via incomplete specs or questions)

Reliability Availability Security Maintainability Portability.

Broadly software requirements should be categorized in two categories:

### **Functional Requirements**

Requirements, which are related to functional aspect of software fall into this category.

They define functions and functionality within and from the software system.

#### EXAMPLES -

* Search option given to user to search from various invoices.
* User should be able to mail any report to management.
* Users can be divided into groups and groups can be given separate rights.
* Should comply business rules and administrative functions.
* Software is developed keeping downward compatibility intact.

### **Non-Functional Requirements**

Requirements, which are not related to functional aspect of software, fall into this category. They are implicit or expected characteristics of software, which users make assumption of.

Non-functional requirements include -

* Security
* Logging
* Storage
* Configuration
* Performance
* Cost
* Interoperability
* Flexibility
* Disaster recovery
* Accessibility

Requirements are categorized logically as

* **Must Have** : Software cannot be said operational without them.
* **Should have** : Enhancing the functionality of software.
* **Could have** : Software can still properly function with these requirements.
* **Wish list** : These requirements do not map to any objectives of software.

While developing software, ‘Must have’ must be implemented, ‘Should have’ is a matter of debate with stakeholders and negation, whereas ‘could have’ and ‘wish list’ can be kept for software updates.

User Interface requirements

UI is an important part of any software or hardware or hybrid system. A software is widely accepted if it is -

* easy to operate
* quick in response
* effectively handling operational errors
* providing simple yet consistent user interface

User acceptance majorly depends upon how user can use the software. UI is the only way for users to perceive the system. A well performing software system must also be equipped with attractive, clear, consistent and responsive user interface. Otherwise the functionalities of software system can not be used in convenient way. A system is said be good if it provides means to use it efficiently. User interface requirements are briefly mentioned below -

* Content presentation
* Easy Navigation
* Simple interface
* Responsive
* Consistent UI elements
* Feedback mechanism
* Default settings
* Purposeful layout
* Strategical use of color and texture.
* Provide help information
* User centric approach
* Group based view settings.

Software System Analyst

System analyst in an IT organization is a person, who analyzes the requirement of proposed system and ensures that requirements are conceived and documented properly & correctly. Role of an analyst starts during Software Analysis Phase of SDLC. It is the responsibility of analyst to make sure that the developed software meets the requirements of the client.

System Analysts have the following responsibilities:

* Analyzing and understanding requirements of intended software
* Understanding how the project will contribute in the organization objectives
* Identify sources of requirement
* Validation of requirement
* Develop and implement requirement management plan
* Documentation of business, technical, process and product requirements
* Coordination with clients to prioritize requirements and remove and ambiguity
* Finalizing acceptance criteria with client and other stakeholders

# **System Requirements**

[System Requirements](http://sebokwiki.org/wiki/System_Requirements)

[**System requirements**](http://sebokwiki.org/wiki/System_Requirement_(glossary)) are all of the [**requirements**](http://sebokwiki.org/wiki/Requirement_(glossary)) at the *system level* that describe the functions which the system as a whole should fulfill to satisfy the [**stakeholder needs and requirements**](http://sebokwiki.org/wiki/Stakeholder_Requirement_(glossary)), and is expressed in an appropriate combination of textual statements, views, and non-functional requirements; the latter expressing the levels of safety, security, reliability, etc., that will be necessary.

System requirements play major roles in systems engineering, as they:

* Form the basis of system [**architecture**](http://sebokwiki.org/wiki/Architecture_(glossary)) and [**design**](http://sebokwiki.org/wiki/Design_(glossary)) activities.
* Form the basis of system [**integration**](http://sebokwiki.org/wiki/Integration_(glossary))and [**verification**](http://sebokwiki.org/wiki/Verification_(glossary)) activities.
* Act as reference for [**validation**](http://sebokwiki.org/wiki/Validation_(glossary)) and stakeholder acceptance.
* Provide a means of communication between the various technical staff that interact throughout the project.

# [**What is the difference between user requirements and system requirements?**](http://programmers.stackexchange.com/questions/264113/what-is-the-difference-between-user-requirements-and-system-requirements)

**User Requirement Definition:**

1. The MHC-PMS shall generate monthly management reports showing

the cost of drugs prescribed by each clinic during that month.

**System Requirements Specification:**

1.1. On the last working day of each month, a summary of the drugs

prescribed, their cost, and the prescribing clinics shall be generated.

1.2. The system shall automatically generate the report for printing after

17.30 on the last working day of the month.

The first example of a "user requirement" is more like a wish or "feature." The way you can tell the difference between a feature and a requirement is that there's enough detail in the requirement to make it *testable.* Requirement 1 is not testable because, well, it's a wish. "I wish that the system had some reports for the managers." How do you know that the requirement has been achieved, that you can declare success?

Requirement 1.1 is testable because you can wait until the last working day of the month, and see if a report is generated on that day (or you can inject dates into the system and observe its behavior).

Requirement 1.2 is testable for the same reasons.

Neither system requirement, however, tells you what the reports should look like, how the data is laid out, or how the calculations are made; they only describe the reports in general terms. In practice, there will be a Software Design Specification of some sort that tells you in detail what these reports will look like.

**User requirements** talk about the problem domain, the world of the user. They describe what effects need to be achieved. These effects are the *combined responsibility of the software, the hardware, and the users*.

**System requirements** talk about the solution domain, the world of the software logic. They describe what the software must do (as opposed to the effects in the user's world that this may or may not achieve).

For instance for a bookkeeping software,

* the user requirement is to compute the correct revenue.
* But the system requirement is only to compute the correct sum of the partial revenues entered by the user.

If the user enters incorrect partial revenues the software is not required to magically correct them: The output will be the correct sum of the inputs, but *not* the correct overall revenue.

The difference is not overly interesting for most simple information systems. It can be very important for life-critical software; see for instance the various accidents involving the issue when or when not the thrust reversal can be activated on a commercial airplane: The user requirement "reverse thrust can only be activated if the airplane is on a runway" has turned out to be [surprisingly tricky to turn into system requirements that reliably lead to the desired effect](https://en.wikipedia.org/wiki/Lufthansa_Flight_2904#Details_about_the_design_features_of_the_aircraft).

|  |  |
| --- | --- |
| 0down vote | [**User requirements**](http://en.wikipedia.org/wiki/User_requirements_document) tell what application must/should do to satisfy user's needs. It a list of features an application must/should have, and it is used a guidance when you develop an application: then all points are checked, you are (probably) done.  I tend to call this simply the "list of features".  [**System requirements**](http://en.wikipedia.org/wiki/System_requirements) tell what system should have to be able to run the program:   * Hardware: CPU, memory, disk space, etc. * Software: OS, libraries, packages, etc.   There is a bit of confusion:   * In "user requirements" the user is a subject, the one that require, and the program being developed is an object. * In "system requirements" the program being developed is a subject (and it's not even mentioned in a phrase), while "system" is an object.   The word "system" can refer to different things:   * software being developed * physical device which runs the software being developed * network of such devices * OS together with execution environment   The thing you've called "*System* Requirements Specification" is probably something like the *[Software](http://en.wikipedia.org/wiki/Software_requirements_specification)*[requirements specification](http://en.wikipedia.org/wiki/Software_requirements_specification) - I consider that to be a superset of user requirements. |

Stakeholder:

A **stakeholder** is a party that has an interest in a company, and can either affect or be affected by the business. The primary **stakeholders** in a typical corporation are its investors, employees and customers.

## Types of stakeholders[[edit](https://en.wikipedia.org/w/index.php?title=Stakeholder_(corporate)&action=edit&section=3)]

Any action taken by any organization or any group might affect those people who are linked with them in the private sector. For examples these are parents, children, customers, owners, employees, associates, partners, contractors, and suppliers, people that are related or located nearby.

**Primary Stakeholders** - usually internal stakeholders, are those that engage in economic transactions with the business. (For example stockholders, customers, suppliers, creditors, and employees)

**Secondary Stakeholders** - usually external stakeholders, are those who - although they do not engage in direct economic exchange with the business - are affected by or can affect its actions. (For example the general public, communities, activist groups, business support groups, and the media)

**Excluded Stakeholders** - those such as children or the disinterested public, originally as they had no economic impact on business. Now as the concept takes an [anthropocentric](https://en.wikipedia.org/wiki/Anthropocentrism) perspective, while some groups like the general public may be recognized as stakeholders others remain excluded. Such a perspective does not give plants, animals or even geology a voice as stakeholders, but only an [instrumental value](https://en.wikipedia.org/wiki/Instrumental_value) in relation to human groups or individuals.