How to write a software requirement document (with template)

JUMP TO SECTION

1. [What is a software requirement specification document (SRS)?](https://asana.com/resources/software-requirement-document-template#what-is-a-software-requirement-specification-document-srs)
2. [Why use an SRS?](https://asana.com/resources/software-requirement-document-template#why-use-an-srs)
3. [What to include in an SRS document](https://asana.com/resources/software-requirement-document-template#what-to-include-in-an-srs-document)
4. [Software requirement document template](https://asana.com/resources/software-requirement-document-template#software-requirement-document-template)
5. [Best practices for writing an SRS document](https://asana.com/resources/software-requirement-document-template#best-practices-for-writing-an-srs-document)
6. [Use software requirement documents to clarify your vision](https://asana.com/resources/software-requirement-document-template#use-software-requirement-documents-to-clarify-your-vision)

Summary

Even if they lack the technical experience, a software requirement document template helps project managers and analysts communicate software expectations with developers. We’ll cover when and how to write one, as well as best practices to ensure your team is working towards the same goal.

Do you remember reading 19th-century novels in school and thinking, “Is this even the same language?” Well, it’s likely you’ve had that exact thought in the office when collaborating with tech-minded AI developers or web-savvy SEO analysts. If only there were CliffsNotes for colleagues.

Sometimes it’s essential for departments on opposite ends of an organization to work together—even if they speak different technical languages. If you’ve ever worked on a [cross-functional team](https://asana.com/resources/cross-functional-team), you know how challenging it can be to keep everyone on the same page.

Software requirement specification documents can help project managers, product managers, and business analysts break down high-level concepts into action items that every team member can follow during the development process.

What is a software requirement specification document (SRS)?

**A software requirement specification document is a detailed description of what a software will do and how it should perform.**

A software requirement specifications (SRS) document lists the requirements, expectations, design, and standards for a future project. These include the high-level business requirements dictating the goal of the project, end-user requirements and needs, and the product’s functionality in technical terms. To put it simply, an SRS provides a detailed description of how a software product should work and how your development team should make it work.

Imagine you have a great idea for an app. You have a vision of what you want it to do and how you want it to look, but you know you can’t just give a verbal description to a developer and expect them to match your expectations. This is where an SRS comes in.

Why use an SRS?

If developers don’t have clear directions when creating a [new product](https://asana.com/resources/product-development-process), you may end up spending more time and money than anticipated trying to get the software to match what you had in mind.

Composing an SRS document helps you put your idea down on paper and set a clear list of requirements. This document becomes your product’s sole source of truth, so all your teams—from marketing to maintenance—are on the same page.

Because software requirement specifications are living documents, they can also act as a communication point between every [stakeholder](https://asana.com/resources/stakeholder-vs-shareholder) involved in the product development process. Product iterations are bound to occur during any software development project—by noting changes in the SRS, all parties can validate them in the document. This will ease any confusion regarding product requirements.

What to include in an SRS document



A basic SRS document outline has four parts: an introduction, system and functional requirements, external interface requirements, and non-functional requirements.

1. Introduction

An SRS introduction is exactly what you expect—it’s a 10,000-foot view of the overall project. When writing your introduction, describe the purpose of the product, the intended audience, and how the audience will use it. In your introduction, make sure to include:

* Product scope: The [scope](https://asana.com/resources/scope-management-plan) should relate to the overall business goals of the product, which is especially important if multiple teams or contractors will have access to the document. List the benefits, objectives, and goals intended for the product.
* Product value: Why is your product important? How will it help your intended audience? What function will it serve, or what problem will it solve? Ask yourself how your audience will find value in the product.
* Intended audience: Describe your ideal audience. They will dictate the look and feel of your product and how you market it.
* Intended use: Imagine how your audience will use your product. List the functions you provide and all the possible ways your audience can use your product depending on their role. It’s also good practice to include use cases to illustrate your vision.
* Definitions and acronyms: Every industry or business has its own unique acronyms or jargon. Lay out the definitions of the terms you are using in your SRS to ensure all parties understand what you’re trying to say.
* Table of contents: A thorough SRS document will likely be very long. Include a table of contents to help all participants find exactly what they’re looking for.

Make sure your introduction is clear and concise. Remember that your introduction will be your guide to the rest of the SRS outline, and you want it to be interpreted the same by everyone using the doc.

2. System requirements and functional requirements

Once you have your introduction, it’s time to get more specific.Functional requirements break down system features and functions that allow your system to perform as intended.

Use your overview as a reference to check that your requirements meet the user’s basic needs as you fill in the details. There are thousands of functional requirements to include depending on your product. Some of the most common are:

* If/then behaviors
* Data handling logic
* System workflows
* Transaction handling
* Administrative functions
* Regulatory and compliance needs
* Performance requirements
* Details of operations conducted for every screen

If this feels like a lot, try taking it one requirement at a time. The more detail you can include in your SRS document, the less troubleshooting you’ll need to do later on.

3. External interface requirements

External interface requirements are types of functional requirements that ensure the system will communicate properly with external components, such as:

* User interfaces: The key to application usability that includes content presentation, application navigation, and user assistance, among other components.
* Hardware interfaces: The characteristics of each interface between the software and hardware components of the system, such as supported device types and communication protocols.
* Software interfaces: The connections between your product and other software components, including databases, libraries, and operating systems.
* Communication interfaces: The requirements for the communication functions your product will use, like emails or embedded forms.

Embedded systems rely on external interface requirements. You should include things like screen layouts, button functions, and a description of how your product depends on other systems.

4. Non-functional requirements (NRFs)

The final section of your SRS details non-functional requirements. While functional requirements tell a system what to do, non-functional requirements (NFRs) determine how your system will implement these features. For example, a functional requirement might tell your system to print a packing slip when a customer orders your product. An NFR will ensure that the packing slip prints on 4”x6” white paper, the standard size for packing slips.

While a system can still work if you don’t meet NFRs, you may be putting user or stakeholder expectations at risk. These requirements keep functional requirements in check, so it still includes attributes like product affordability and ease of use.

The most common types of NFRs are called the ‘Itys’. They are:

* Security: What’s needed to ensure any sensitive information your software collects from users is protected.
* Capacity: Your product’s current and future storage needs, including a plan for how your system will scale up for increasing volume demands.
* Compatibility: The minimum hardware requirements for your software, such as support for operating systems and their versions.
* Reliability and availability: How often you expect users to be using your software and what the critical failure time is under normal usage.
* Scalability: The highest workloads under which your system will still perform as expected.
* Maintainability: How your application should use continuous integration so you can quickly deploy features and bug fixes.
* Usability: How easy it is to use the product.

Other common types of non-functional requirements include performance, regulatory, and environmental requirements.

Software requirement document template

Ready to start your own software-development venture? Our SRS template outlines all four key components of a great SRS document, giving you and your team valuable insight into the product you will develop. Remember to keep your requirements detailed, clear, and concise, so all parties have the same vision in mind.



Best practices for writing an SRS document

The purpose of an SRS is to keep each team in every department working towards a clear goal. That being said, there are a few best practices to follow to ensure your SRS serves its purpose.

Enrich your SRS with visuals

Including visuals like diagrams, schemes, and models will help team members better understand the process. These are especially useful when illustrating the main functions and operability of your software.

One technique to try while brainstorming your project is mind mapping, which organizes ideas, features, and scenarios and draws the connections between them. Create a mind map to structure random thoughts as you begin to piece together your ideas. This visual doesn’t need to be super detailed—that’s what your SRS is for. Instead, focus on the key functions of your software and how they relate to one another.

[Read: 29 brainstorming techniques: effective ways to spark creativity](https://asana.com/resources/brainstorming-techniques)

Keep it clear and concise

The last thing you want is your developers second-guessing themselves when constructing your product. Try not to leave room for team members to get creative and fill in the blanks. Include as much detail as possible when describing your software requirements, and avoid:

* Using vague words like *generally* or *approximately*
* Combining terms with a “/”, which could be interpreted as “and” or “or”
* Using complicated boundary values
* Using double and triple negatives

A formal peer review is a good way to pinpoint ambiguities in your SRS document. Plan to go over it with each participant to compare his or her understanding of the requirements and make the necessary changes.

Know your end-user

Add your field research and user interviews in the SRS to construct a clear understanding of your end-users requirements, expectations, and needs. This should help you visualize the operations your end-user will perform with the software. Take into account every possible scenario and nuance that could happen and include it in your SRS. Remember, your developers will implement exactly what you include in the document—no more, no less.

Include a margin for flexibility

Your SRS is a living document, meaning you will add new features and modifications with every iteration. Account for that by keeping requirements flexible in case the outcome doesn’t meet your expectations. It’s also good practice to keep a record of the changes made to the document to avoid any misunderstandings. Participants should be able to trace each requirement to its original and see who makes the change, when, and why.

Use software requirement documents to clarify your vision

Writing an SRS is not easy—but neither is endless troubleshooting or navigating arguments amongst your team members. The work you put into a comprehensive software requirement specifications document will pay off with a stunning product you and your stakeholders can be proud of.