WHITE PAPER

9 Tips for Writing Useful Requirements

If you work for a company that has explicitly defined standards for writing product requirements, consider yourself lucky. However, if you're like the rest of us and don't have the benefit of documented standards, the following nine tips can help you write clear, concise, useful requirements.

We've also included a handy checklist that will help ensure you always write good requirements.

1. Know Your Audience

Keep your audience in mind when writing requirements, and provide the appropriate information and terminology for them. Is your audience technical? Are they team members or customers? How much background about the subject matter are you assuming they have? How about the history of the project? Do you need to include links to previous requirements or related documents and artifacts?

It could be that your primary audience (e.g., the team developing the product) doesn't need much background information, but your secondary audience (e.g., the customer) might. If that's the case, write to your primary audience, but provide links to supporting material for your secondary audience to get up to speed.

Knowing your audience means you'll be able to make certain assumptions about existing knowledge, which in turn makes you better able to determine the right amount of detail to include, the vocabulary to use, and so forth.

2. Provide Relevant Information

Once you understand your audience, make sure you know what the requirement will be used for: to provide enough details for the developers and testers, to get management and stakeholder buy-in, etc. The requirement should provide the relevant information the audience needs.

Different types of requirements have different content needs; high-level business requirements include different information than lower-level technical specifications, for example. From a QA perspective, this means all functional requirements or user stories should be verifiable (i.e., they clearly describe the inputs and expected output). From a development perspective, it means requirements should be feasible (i.e., they clearly describe functionality that can be implemented).

Bad Requirement	Better Requirement
The system shall provide quick response times to user requests.	The system must provide response times within 0.25 seconds of user-initiated requests.
As a WYSICORP customer, I need to save my order so that later I can save a copy, print, or email the list for other uses.	As a WYSICORP customer, I need to save, copy, print, and email my order so that I can edit it again, check a received shipment against a printed list, and send the list to a supplier.
The system must log errors when logins fail.	The system must add an error message to the Server log when users attempt to login with an invalid username or password.

Knowing your audience means you'll be able to make certain assumptions about existing knowledge, which in turn makes you better able to determine the right amount of detail to include, the vocabulary to use, and so forth.

3. Use Clear, Simple Language

Useful requirements are clear, concise, simply worded, and descriptive. Remember, people have to read these, not machines; write them in a way that everyone reviewing the requirement document can understand them. Replace complex words with simple ones. Avoid made-up words and keep your writing jargon-free.

Bad Wording
The system shall enforce the usage of a robust password protocol by disallowing any user password not consisting of, at minimum, eight alphanumeric characters, of which there must be, at minimum, one uppercase character and one numeric character.
Better Wording
User passwords must be at least eight characters long, and contain at least one number and one uppercase letter.

4. Find The Sweet Spot

Aiming to write shorter requirements encourages you to be more precise, and it makes the requirement easier to review, estimate, and mark as being in or out of scope. That said, you don't want to make them too short. You need to find the sweet spot that provides enough — but not too much — detail.

When determining how much information to include in a requirement, consider the complexity of the project, the methodology being used, and any regulatory requirements for documentation.

Requirements generally need more details if the team is spread out geographically and includes contractors in another country, stakeholders in another office, technical resources who are often traveling, or any other team members who lack the opportunity to get face-to-face clarifications.

When determining how much information to include in a requirement, consider the complexity of the project, the methodology being used, and any regulatory requirements for documentation.

5. Take A Picture

Just as "a picture is worth a thousand words," one way to provide enough detail without writing too much is to include diagrams or mock-ups to show what the screens or dialogs should look like. It can be particularly helpful to include an interactive prototype or wireframe to validate requirements with end users and stakeholders; such models make it easier for them to get a feel for the feature.

The danger, though, is that mock-ups and screenshots can quickly become out of date. Make sure to clearly note any out-of-date screenshots in the requirement, and make a plan to update them when necessary.

6. Be Consistent

Requirements should be written in a consistent format or style, and grouped together in a logical order in the requirements document. This helps reviewers easily understand how the pieces fit together.

Create a checklist or template to help requirement writers make sure they include the right information.

7. Establish Clear Ownership

Make sure everyone on the team knows who is responsible for writing each requirement and who has the final say in approving or rejecting it. A simple table added to your project documentation is all you may need. All team members should have access to this information.

It's also helpful to have a detailed change history, so when questions arise or confusion sets in, it's obvious who can provide clarification.

8. Share The Status

Your project management process should include a way to communicate the status of each requirement to the team.

- Is the requirement still in the draft stage?
- Is it ready for review, out-of-date, or obsolete?
- Has it been implemented or approved?

Ideally, you want status updates to be automated so team members don't have to spend time compiling and emailing information to each other.

Requirements should be written in a consistent format or style, and grouped together in a logical order in the requirements document. This helps reviewers easily understand how the pieces fit together.

If requirements status isn't visible to the team, developers and QA may start working with requirements that are still being refined. It's frustrating for requirement writers (business analysts), and can cause errors and re-work.

9. Listen To Feedback

Conduct a postmortem after each release, and examine the issues found during the project. Gathering this kind of feedback tells you how well current requirement styles are working. It can help you identify what worked, so that you can repeat it the next time around. More importantly, it will help you identify what didn't work and prevent you from wasting time and effort in the future.

Pay special attention to the bugs written up by QA because the functionality didn't match their expectations. This could indicate ambiguous requirements, outdated test cases, or other requirements-related process issues.

Checklist: Ensuring Good Requirements

☐ **IS THE REQUIREMENT COMPLETE?**
- Is the requirement written as a complete sentence?
- Can the reader understand the requirement without having to reference additional information?

☐ **IS THE REQUIREMENT CLEAR?**
- Is the requirement unambiguously worded?
- Do all stakeholders agree on the meaning of the requirement?

☐ **IS THE REQUIREMENT CONSISTENT?**
- Does the requirement conflict with other requirements?
- Is the terminology used consistent with other requirement and glossary terms?

☐ **IS THE REQUIREMENT VERIFIABLE?**
- Has the testing or QA team reviewed the requirement to ensure it can be tested?
- Can the requirement be verified in other ways (e.g., inspection, analysis, or demonstration)?

☐ **IS THE REQUIREMENT TRACEABLE?**
- Is the requirement uniquely identified so that it can be referenced clearly?

☐ **IS THE REQUIREMENT DESIGN INDEPENDENT?**
- Does the requirement impose constraints on the design? If so, are the constraints justified?
- As it is written, can the requirement be met in more than one way?

About Perforce

Enterprises across the globe rely on Perforce to build and deliver digital products faster and with higher quality. Perforce offers complete developer collaboration and agile project management tools to accelerate delivery cycles – from agile planning tools to requirements, issues and test management, which then link to all source code, binary assets and artifacts for full build and release tracking and visibility. The company's version control solutions are well known for securely managing change across all digital content – source code, art files, video files, images, libraries - while supporting the developer and build tools your teams need to be productive, such as Git, Visual Studio, Jenkins, Adobe, Maya and many others. Perforce is trusted by the world's most innovative brands, including NVIDIA, Pixar, Scania, Ubisoft, and VMware. The company has offices in Minneapolis, MN, Alameda, CA, Mason, OH, the United Kingdom, Finland, Sweden, Germany, and Australia, and sales partners around the globe. For more information, please visit www.perforce.com