# HOW TO WRITE AN EFFECTIVE PRODUCT REQUIREMENTS DOCUMENT

by Damien Filiatrault | CEO January 18, 2017

Any aspiring Product Owner looking to build a great software product could be forgiven for feeling overwhelmed. A quick Google search turns up a lot of conflicting, dated examples for a product requirements document. People used to follow the Waterfall Model and define everything their software would do at the outset (think bloated Use Cases and UML diagrams).

We don't want to waste precious time trying to define every possible thing your software will do and frankly, no one likes writing (or reading) a verbose product requirements document.

Today, savvy project managers have shifted to the new, Agile side of the spectrum where the product is released quickly, feedback is obtained and improvements are made iteratively. While I am a big believer in Agile and Scrum, it is not sensible to just hire a development team and start building features without knowing what you are getting into.

This article will help you write a product requirements document that will be good enough to get a reasonably accurate estimate from a dev shop. We want to strike the right balance between being prepared and being agile. In other words, the bare minimum effort needed to get started building a great product.

## WHAT IS A PRODUCT REQUIREMENTS DOCUMENT?

A Requirements Document should act as the starting point for your product: outlining its purpose, who will use it, and how to use it. It is an essential precursor to design and development.

This article should help you create a requirements document that straddles the line between concise and precise.

This article is for entrepreneurs and product owners who are looking to build a digital product. To avoid sounding too vague I have decided to use examples from an e-commerce website, but this could easily apply to creating a mobile app or software program.

## THE 'BARE MINIMUM' FRAMEWORK

Often a product owner will have grand ideas for their project and all of the features it might have in the future. Dreaming big is great, but it's extremely important at the outset to narrow your focus down to the minimum set of features that you need in the first release. This is called the Minimum Viable Product. After you release your product and get feedback from your users, you can then decide what additional features you want to add. But while you are writing your product requirements document, clear your head of all those potential future features and just define those that will be included in the first version of your product.

*Note: Of course, if there is a future feature that must be known about at the outset, then the development team should know it is coming and I'll talk about how to include that.*

Enough preamble, below are the sections I suggest for a simple product requirements document:

1. Goals
2. User Personas
3. User Stories
4. Sitemap
5. Page Descriptions
6. Wireframes (optional)
7. Non-Functional Requirements
8. Risks
9. Future Iterations

## GOALS

The business goals and objectives serve as the context, showing the dev team why they are building what they are building.

Here are some common questions you can ask yourself when fleshing out this section:

- What is the purpose of this project?
- What are the problems it will solve?
- How will it streamline or improve the current process or facilitate a new process?
- What is the product vision?

Check out our Free Product Requirements DocumentTemplate for more examples.

## USER PERSONAS

User Personas are hypothetical individuals who match your desired, or actual, audience. Thinking about the background of these users will improve your ability to create a product that meets their needs.

User personas are one of the areas that people tend to drown in. So let's establish some rules to keep you afloat.

- Cover the primary types of users. You don't need to create a persona for every kind of user, just enough to illustrate the main groups who will use your product. For simple applications, 3 good profiles will cover more than 80% of your user base, but if your application is complex you may need more.
- Focus on what you know. Making up details to simply fill an end user profile is counterproductive.
- Start with these 5 metrics. Occupation, age, gender, location, education. They will be the framework for your profile.
- Name your End Users. It's easier to relate to "David the Executive" than to "End User D".
- Give each profile an objective. Each user needs a goal that fits their profile. This will be their raison d'etre.

Essentially it boils down to this: your personas should be detailed enough to allow you to see the product through their eyes. Each profile should function like another person in the room when making a decision. "Would David login to read his message or should we email it to him?"

If you have an existing website, use Google Analytics to help you create these profiles. It gives you invaluable information, you can see where your visitors came from, what keywords they used to find you, and what content and behaviors they exhibit online. If this is a new product, Alexa (and other similar sites like Similar Web) will give you similar data for your competition.

## USER STORIES

User stories are short descriptions of a feature, told from the perspective of one of your newly created end user profiles. They are typically structured in the following fashion:

**As a [type of user], I want [some goal] so that [some reason].**

User stories are a starting point, not a destination. It's often best to think of them as pointers to your eventual requirements. These stories are vital because the discussions they start will help shape your content architecture and design.

So how can you use this concept to move your product forward without getting bogged down in hundreds of stories and feature ideas?

- Limit yourself to high level (and must have) stories. These high level stories are known as Epics and can be broken down later into smaller more manageable stories.
- Your stories should be short and specific.
- They should describe who needs what and why
- They must be user-centric. Remember you can use your personas.

Here is an example focussed on a user role:

- As an Admin, I want to be able to see new products and products categories as they are ordered so I can write content for the website.

And here is one more focused on the user persona:

- As a Website User, making the correct product choice is vital. I want to see my product options side by side so I can make an educated choice quickly.

You don't have to write a user story for every little bit of functionality in your application up front. You should focus on the Epics because details will be fleshed out when wireframes and designs are created. Epics are broken down into smaller stories in backlog grooming sessions. Remember, at this point, you are writing to give the designers and developers the minimum information needed to start a productive conversation about how to create your application.

## SITEMAP

Once your main user stories are defined, you should have a solid idea of what pages or screens your application will page. The sitemap is the first step to documenting this.

A good sitemap includes the following:

- A comprehensive list of all pages or screens. This includes everything from pages detailing information about products and services to an "About Us" page and a privacy policy. Spend some time brainstorming to ensure everything you want in your application is listed.
- The hierarchy between these pages (optional). If you have an idea of how your pages relate to each other hierarchically, then communicating this will be helpful for designing the navigation. However, this step is not a must, and a comprehensive list of all pages is enough for the bare minimum.

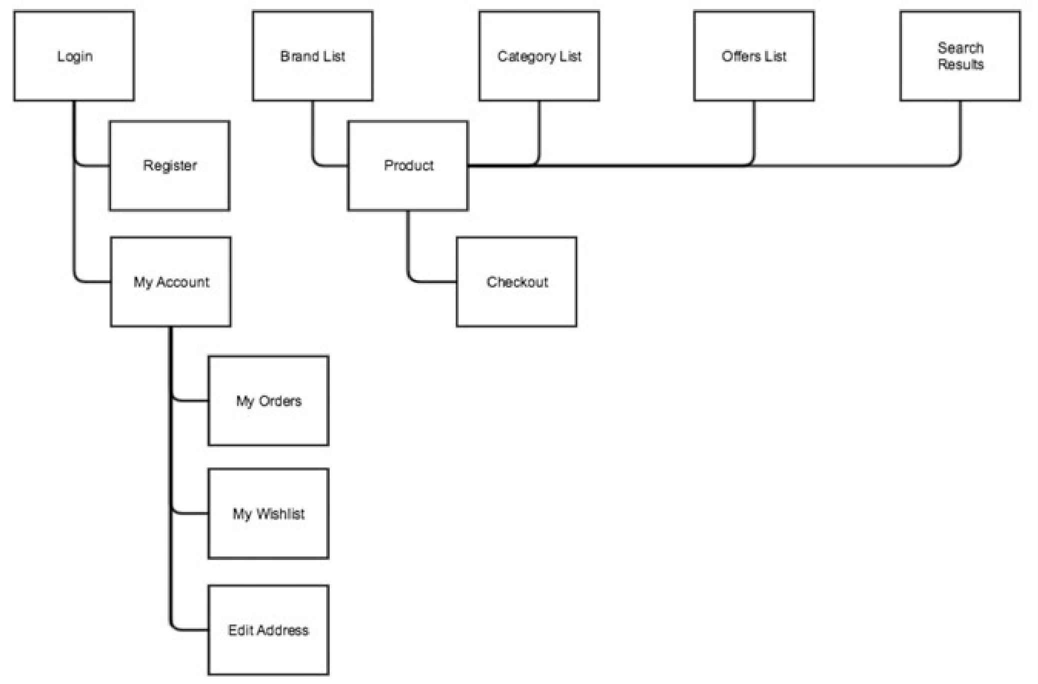
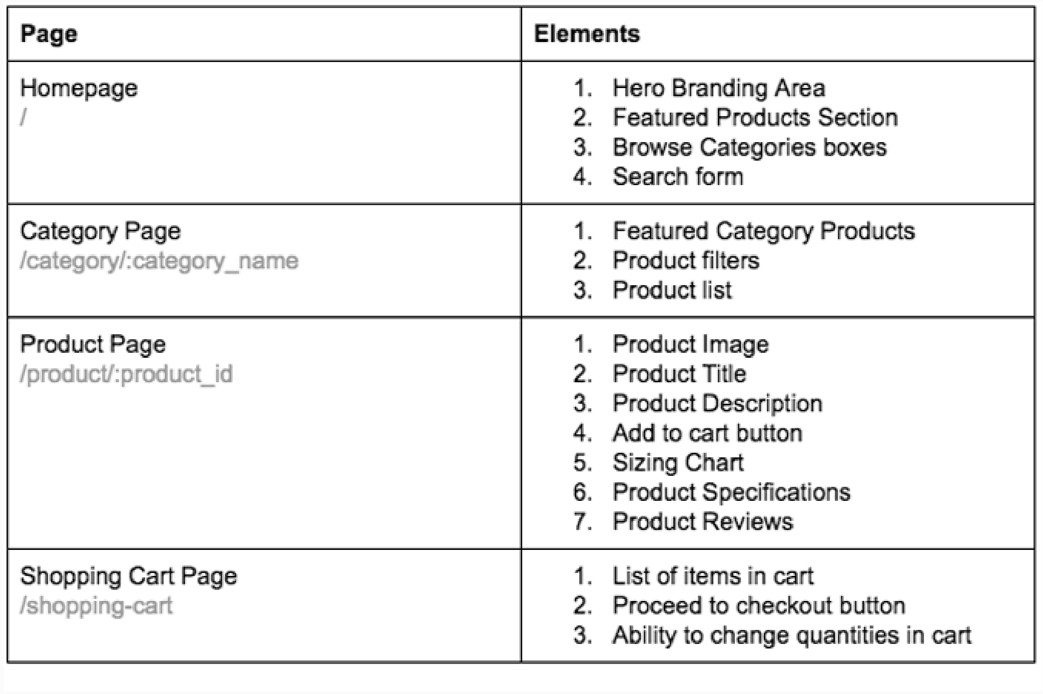


*Figure 1: Example Sitemap*

A good solution to quickly create a sitemap that you can easily share with your stakeholders isGloo Maps. Or just use one of the free add-ons for Google Docs.

## PAGE DESCRIPTIONS

Once you have a list of all of the pages or screens in your application, you can take the next step and define what will be on each page. Make a simple numbered list of the major items that will be included on each page and put the most important items at the top of the list (this will convey the relative importance of each item which will inform the design). Here are some simple examples from an e-commerce website.



| **Page** | **Elements** |
|---|---|
| Homepage<br>/ | 1. Hero Branding Area<br>2. Featured Products Section<br>3. Browse Categories boxes<br>4. Search form |
| Category Page<br>/category/:category_name | 1. Featured Category Products<br>2. Product filters<br>3. Product list |
| Product Page<br>/product/:product_id | 1. Product Image<br>2. Product Title<br>3. Product Description<br>4. Add to cart button<br>5. Sizing Chart<br>6. Product Specifications<br>7. Product Reviews |
| Shopping Cart Page<br>/shopping-cart | 1. List of items in cart<br>2. Proceed to checkout button<br>3. Ability to change quantities in cart |

*NOTE: If you want to go the extra mile, you can define the URL of each page which often helps to organize things.*

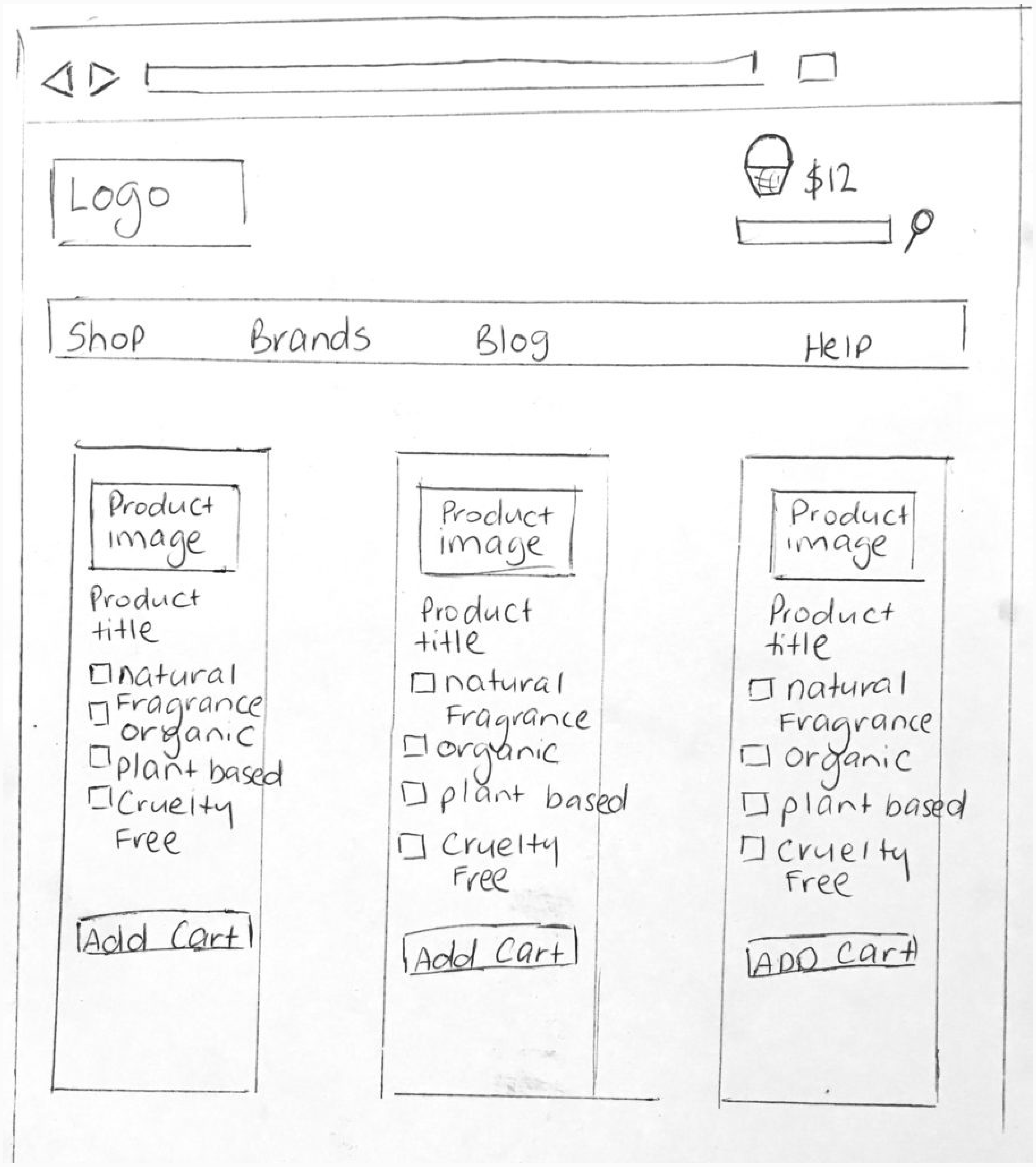
## WIREFRAMES

Wireframes are simple page layouts that outline the size and placement of elements, features on a page. They are generally devoid of color, font styles, logos or any design elements.

Think of a wireframe as a blueprint that shows you the location of elements set apart from the design of those elements. Issues will often surface that may be only noticeable via a visual tool. I often find myself moving or even removing, features of a website when I get to the wireframing stage.

Due to the iterative nature of the process, I will sketch the first few attempts onto a notepad and then move into a simple prototyping tool like Balsamiq. The figures below show you the progression of my wireframes from paper sketches to a final version.

*Figure 2: A Sketched wireframe of the product comparison table on the website*

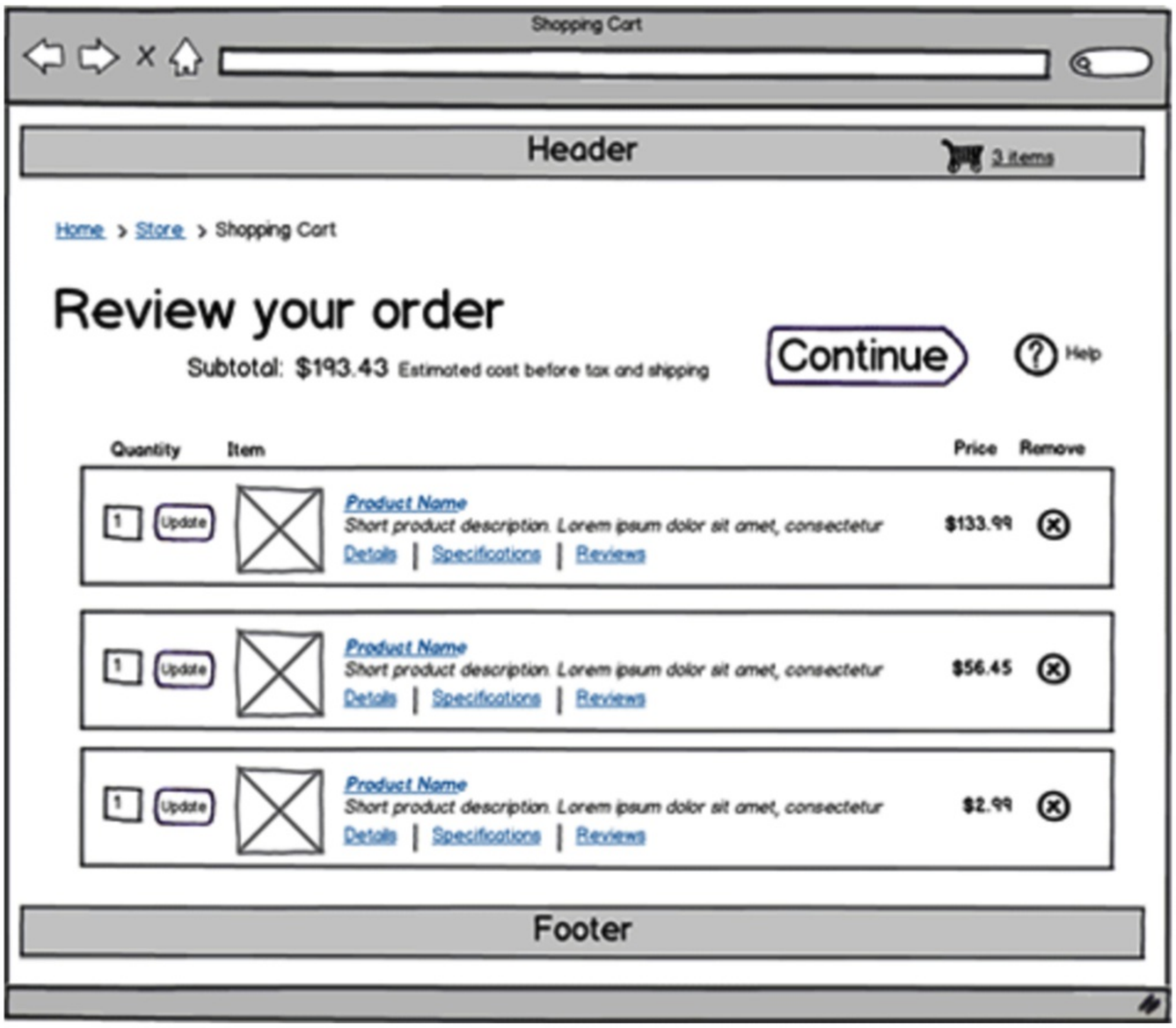


Wireframing is probably the most time-consuming step of this process and for some simple projects, it may be overkill. For complex projects where serious design thinking needs to happen, wireframes are an indispensable tool.

As the Product Owner, you may not have experience with wireframing, but I would encourage you to try doing it yourself. Doing sketches on paper and taking a picture with your phone can work remarkably well. I find that when entrepreneurs have taken the time to learn how to wireframe and deeply think about how their product will work, their development projects have gone more smoothly.

That said, wireframes are an optional part of the 'bare minimum' framework because it can be shifted to design phase of a project. When a Product Owner provides us with all of the other information in this article, we have enough information to prepare a reasonably accurate estimate of the effort it will take to build the product and get started.

*Figure 3: A sitemap created in prototyping software.*


Shopping Cart
Header
3 items
Home > Store > Shopping Cart
Review your order
Subtotal: $193.43 Estimated cost before tax and shipping
Continue
Help
Quantity
Item
Price
Remove
Update
Product Name
Short product description. Lorem ipsum dolor sit amet, consectetur
Details | Specifications | Reviews
$133.99
$56.45
$2.99
Footer

## NON-FUNCTIONAL REQUIREMENTS

While the bulk of the product requirements document defines how the software will function (functional requirements), this part of the document defines requirements that may be important to your business, but are not about how the software itself functions. This is a place where you can communicate any special parameters that the developers will need to take into consideration. Here are some examples:

- The application must be built in Ruby on Rails
- The application must be hosted on AWS
- The application must use Stripe for payment processing
- The application must work in all modern browsers
- The application must be responsive (work well and look good on all screen sizes)
- The application must be able to support 1000 simultaneous users

## RISKS

If there are any significant, known risks that the project faces, it is a good idea to document them. Frequently, it is a good idea for a development team to try to tackle risky parts of a software project first so you can know early on before you invest too much time and money if a particular feature is not feasible.

Here are some examples:

- Our predictive recommendation engine, which is a key differentiator for our start-up, may be difficult to code.
- Our business account may not be approved with Stripe.

## FUTURE ITERATIONS

While you are writing the product requirements document, you should constantly be thinking *"Is this absolutely necessary for my MVP?"*. If a feature is a good idea and important to the long-term plan for the business, but doesn't make it into the MVP, you should document it briefly in the future Iterations section so that it is not forgotten and developers can make sure the application can be extended in the future to support these key features.

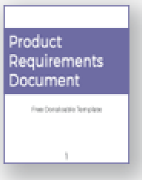
## CONCLUSION

Hopefully, this article has demonstrated how a good product requirements document will save both time and money throughout the life of a project. To further speed up the process for you, we've included a link to download the actual Document Template we use at Scalable Path. I've also added some useful examples that tie in with this article.

The easiest way to edit your own version of our Template is to go the top menu and select 'File' and 'Add to Drive'. This will then save a copy of the document to your Google Drive.

Alternatively, you can select the 'Download As' option.

As a final note, not every client has the time to create a requirements document on their own and that's OK. We're happy to take on this process on by engaging in a brief discovery phase to help you define your requirements.



**We have created an Example Requirements Document to help you quickly launch your next project.**

View the Document on Google Drive here.

**Are you looking for help with your next software project?**
You've come to the right place, every Scalable Path developer has been carefully hand picked by our technical recruitment team. Contact us and we'll have your team up and running in no time.

*Agile, Product Owner, Product Requirements Document, Sitemaps, Software, User Personas, User Stories, Waterfall, Wireframes*

## POPULAR ARTICLES

7 Qualities That Differentiate a Great Programmer from a Good Programmer
*posted on*
*09/03/2015*

How to Write an Effective Product Requirements Document
*posted on*
*01/18/2017*

Using Quill.js To Build A WYSIWYG Editor For Your Website
*posted on*
*12/27/2017*

## ABOUT THE AUTHOR

Damien is the founder of Scalable Path and also acts as an architect and consultant on many of the company's projects. Previously, he headed PHP development at SolutionSet, where he spent a 5 month period in Goa, India managing a team of software developers. He has also held sales and marketing positions at other San Francisco technology companies including Evite and CNET Networks. With bachelors degrees in geography and computer science from UC Berkeley and San Jose State University, Damien brings a unique perspective on international business to both Scalable Path and its clients. He is passionate about agile processes and still enjoys getting in the zone and coding when he can.

INQUIRE ABOUT DAMIEN

## CATEGORIES

Backend Design Development DevOps Frontend Hiring Marketing
Mobile
News & Events Productivity Project Management Remote
Work Strategy Testing Uncategorized

## STAY UP TO DATE

Subscribe to our newsletter

EMAIL

SIGN UP

## LATEST TWEETS

Tweets by @ScalablePath

## GET STARTED WITH SCALABLE PATH

HIRE TOP TALENT APPLY AS A FREELANCER

## BROWSE TALENT

### .NET Developers

ASP.NET (MVC) Developers
ASP.NET (Web API) Developers
C# Developers
WPF Developers
WWF Developers

### Designers

Brand Design
Creative Direction
Graphic Design
Logo Design
Package Design
UI/UX Design
Wireframing

### DevOps Engineers

AWS Engineers
Ansible Engineers
Apache Engineers
Azure Engineers
Chef Engineers
Digital Ocean Engineers
Docker Engineers
Elasticsearch Engineers
Google Cloud Engineers
Grunt Engineers
Gulp Engineers
HAProxy Engineers
Heroku Engineers
Kubernetes Engineers
Linode Engineers
Linux Engineers
Nginx Engineers
Puppet Engineers
Rackspace Engineers
Ubuntu Engineers
Vagrant Engineers

### Frontend Developers

CSS Developers
HTML Developers
LESS Developers
SASS Developers
SCSS Developers

### Java Developers

Java Developers
Dropwizard Developers
Groovy Developers
Play Developers
Spring Developers
Spring Boot Developers

### JavaScript Developers

JavaScript Developers
Angular Developers
Backbone Developers
D3.js Developers
Ember Developers
Express Developers

Flux Developers
MEAN Developers
Meteor Developers
Node.js Developers
React Developers
Redux Developers
TypeScript Developers
Vue.js Developers

### PHP Developers

PHP Developers
Cake PHP Developers
CodeIgniter Developers
Craft CMS Developers
Drupal Developers
Laravel Developers
Lumen Developers
Magento Developers
Symfony Developers
Wordpress Developers
Yii Developers
Zend Framework Developers

### Python Developers

Python Developers
Bottle Developers
Django Developers
Flask Developers

### Ruby Developers

Ruby Developers
Rails Developers
Sinatra Developers

### Mobile Developers

Android Developers
Cordova Developers
Ionic Developers
Kotlin Developers
Objective-C Developers
Phonegap Developers
React Native Developers
Swift Developers
Windows Mobile Developers
Xamarin Developers
iOS Developers

## MORE

Question Marketplace
Client FAQ
Contractor FAQ
Contact us

## STAY UP TO DATE

Join thousands of subscribers already getting our original articles about software design and development. You will not receive any spam, just great content once a month.

EMAIL

SIGN UP

## FOLLOW US