Functional Specification Documents

A tutorial for creation and process

By: Allen W. Smith

Contents

Introduction

What Is A Functional Specification?

Functional specifications (functional specs), in the end, are the blueprint for how you want a particular web project or application to look and work. It details what the finished product will do, how a user will interact with it, and what it will look like. By creating a blueprint of the product first, time and productivity are saved during the development stage because the programmers can program instead of also working out the logic of the user-experience. It will also enable you to manage the expectations of your clients or management, as they will know exactly what to expect.

Why write a Functional Spec?

A key benefit of writing up a Functional Spec is in streamlining the development process. The developer working from the spec has, ideally, all of their questions answered about the application and can start building it. And since this is a spec that was approved by the client, they are building nothing less than what the client is expecting. There should be nothing left to guess or interpret when the spec is completed...and this, in a nut, explains my love affair with the Functional Spec.

Who writes a Functional Spec?

The functional spec should be written by someone who is not involved in any other aspect of the project. You will want somebody who is very familiar with user-interface issues and web design, familiar enough with technology to know its limitations and capabilities, and someone who is a very skilled and detailed writer. While writing a spec, you will spend much of your time imagining how a user might use a certain feature and how they may navigate their way through the information. Not only do you need to map this world out visually, but you also have to write out in great detail what this world does; all the while, balancing everything within the current technological limitations and business demands. The functional spec writer's sole concern is marrying the user-experience with the various departmental, business, and technical requirements of the project.

Define The Application

The information gathering process is the critical step of any successful functional spec. Just as important as the finished document is the thinking process you have to force yourself through in order to begin writing. It makes everyone think about what they are building, why they're building it, who will be using it, how they'll be using it, and what it will end up doing. At this early stage, everyone may have various shades of ambiguity about what they're getting ready to build and it's unlikely that anyone is in total agreement about what exactly the finished product will do. Here is a quick checklist of the general questions you should be asking (and depending on what the project is, you'll probably come up with more specific questions) at this early stage:

What is the application supposed to be?

Pretty basic but critically important. Make sure you have a strong grasp on what the product is before you start anything; additionally, make sure the team you're working with, and the management team ultimately responsible for the product, also have a strong grasp on what you are all doing. Expect lots of arguing and discussion during this stage!

What is the application supposed to do?

Now that you know what the product is supposed to be, you should start dissecting what it is supposed to do. Identify the main objectives of this project based on its mission critical factors. Typically, there are one or two core pieces of functionality that make up the bulk of the final product with a bunch of other smaller items thrown in. Identify those core pieces and make them a priority. Do not allow the smaller items to get in the way, but make sure you've made a note of them. A good way to do this is to simply create a list of all the functions you want the application to perform and then rank them based on priority.

Who is going to be using this application?

Find out who the audience is (i.e., who is going to be using this product). Knowing your audience is important in understanding why and how they are going to be using the final application...which is important in defining the scope of the project. Better yet, create use cases and develop user personas. User personas can be a great tool for allowing you, as the writer, to enter the world of your users and see everything from their perspective.

What are the metrics?

Most projects have a performance metric attached to them somewhere; e.g., revenue, page views, new members, etc. Make sure this is clearly known from the beginning. It will help focus your approach and the approach of the team as well, because even if you develop a great product it can still be a failure if it doesn't meet its metrics.

Is there a precedent for this application?

If what you're developing is similar to other products out there, it can make your learning curve much easier by researching those applications and discovering what works and what doesn't. Make it a point to perform a very thorough competitive analysis. Not only will it help you gain speed faster, but it will also give you an idea of what your competition is like and how to do it better.

***** "Application" is a generic word I'm using for any number of possibilities: a website, a web based tool, a web promotion, etc.

Develop Models

Developing models is a good way to start distilling all of the information you have gathered so far. Models, by definition, are a good way to convey an understanding of the components that make up a system. Teachers frequently use models as aids to explain complex ideas; physicists use atomic models, automobile designers frequently build many models of a new vehicle before creating a prototype. As the functional spec writer, you are responsible for creating information models to help convey the concepts of the application you are developing.

It may be useful to consider creating three different models:

User's Conceptual Model

Building this model will be helped greatly by your development of use cases and user personas. Here, you will build a model of your system based around the user's perceptions of the system: not what the system actually is, but how it will be perceived by the user.

A good example of this is take an e-mail list or a "Clubs" application (like Yahoo! Clubs). The user does not really care whether it is an e-mail list, a club, a web page, or anything. All they care about is the subject: whether that be baseball, german polkas, or Mad magazines. The user's model of the system doesn't incorporate things like "text fields" or "recursive paths," they're only concerned about the subject or information that is most directly related to them.

Designer's Model

The designer's model is really the nuts and bolts model for the functional spec writer. This is where the interface components and relationships to be seen and experienced by the user are defined. It details the available objects in the user's universe and how they can use them to accomplish certain tasks.

It's easier to try and break things out into objects and classes at this point. I find that when I start doing this, the whole system becomes simpler and less complex. Although this topic is much larger than the scope of this tutorial, we can briefly take a look at an example designer's model.

- **View Example Model**
  This example was created in Visio and represents a "User Account" of the example application used throughout this tutorial. See Appendix.

Programmer's Model

The programmer's model is typically only relevant to the programmer. Ideally, we would create user and designer models, and then pass those off to the programmer who would then build the application. However, if you have done this any number of times you will soon realize that every programming environment has inherent limitations and that those constraints must be incorporated into the designer's model. What typically works for me is to have a meeting with the programmer assigned to the project and go over the designer's model with them. From that, I can usually walk away with a good idea of what we need to change, what we can work around, and what I need to watch out for as we move forward.

The most important model for us to consider, as functional spec writers, is the designer's model. This will be the one we spend the most time on and the one that we refer to the most when explaining the application to other stakeholders in the project. It will also form the foundation for writing the spec.

Define The Information Flow

To start this process, try and make an outline of everything you currently know. Since this tutorial is geared towards web projects, make a list of all the pages you know will be included at this point. Underneath your pages, start listing all of the elements, to date, to be included. For example, you could start listing search fields, login buttons, dynamic and static content, etc. Then make this your beginning table of contents. A lot of this should be easy after creating your models for the application.

Three specific items you'll want to attack:

Define the navigational elements

Start creating the "buckets" in which your application is going to be divided up and then identify what is going to be filling up those buckets. Do this from the user's perspective. Seek out what will be your global navigation and how the sub-navigation will work under that to support your user's goals.

Diagram the organization of the information (i.e., create flowcharts)
Creating flowcharts can be very handy when trying to work through a lot of information or a very deep navigation set. Anything you develop now will change over time, but ultimately you'll want a thorough flowchart of the application included with your spec. Also include the navigational path through the information on this diagram (e.g., parent-child directories, where the user can jump to from each page, etc.). Visio is a good tool for developing technical diagrams and flow charts such as these.

View a sample flow diagram
See Appendix

Create a prototype

A prototype of a web application is just a set of static html pages put together to show the key pages of the application and the user interface. It allows the functional spec writer to have something close to a working model to test their ideas out on, start focusing on the layout, and begin gathering input about the look and feel. It is debatable in some circles whether you want to create a prototype at this stage or not. For me, I prefer to have one at this point. In the iterative process, I believe it helps to have this big picture look at a potential finished product while also seeing how the components of the system affect the overall product. Instead of boxes and arrows, bulleted comments, and a list of functions, you have an actual web page to look at and experience. Also, as a web designer I have the capability to do this; which, I realize not every spec writer will have. Whether you create a prototype now or later, you should definitely have one before you start writing the actual spec. Aside from being a convenience for you and allowing you to possibly iterate your design much faster, it allows your colleagues and stakeholders the opportunity to look at the application and deliver much more specific and useful input about what you are doing. Not everyone can look at an informational flowchart and realize something is way they want it. Most people can when looking at a prototype.

Essentially, what you are doing at this stage is defining the relationships of the objects in your designer's model. You are laying the groundwork for what will become the structure of your application.

Create Wireframes And Mockups

The amount of time it takes for you to get to this point is totally dependent on the scope of your project and your team. So far, you've had to spend a lot of time talking to a lot of different people to get input on the various requirements for the application. Don't get frustrated if it seems like it's taking a long time or if it becomes tedious (which it will!). Writing the functional spec imposes a lot of discipline on the whole web development cycle and, in turn, can require a good deal of patience and diligence upfront to make it through. The time spent now, though, will reap multiple dividends later. In terms of development time, frustration, and money saved by having a document that spells out

exactly what the application is and how it should work, it's a small price to pay at the beginning to avoid a death march.

Now that we have defined our application and mapped out the functionality and information, we're ready to create some mockups. At this point, it is recommended you bring in a graphic designer to develop these for you. Even if you're capable of putting them together, it can be beneficial to have someone else looking through your document and creating these. They'll spot holes in your logic and raise good questions; plus, it will free you up to continue focus on the writing and any final requirements gathering you need to do. A basic process you may want to follow:

Create wireframes for your key pages*

A wireframe is a mockup of the page that only addresses the layout, not aesthetics. Think of it as the skeleton of your page, and the mockup will add the skin. A wireframe is useful because, at this point, you're still dealing with pure information and flow but you're also starting to understand the relationship of that information as it applies to your application. The spec writer should be creating these.

- **View an example wireframe**
  See Appendix

Design for the functional requirements, business requirements, and user requirements.
Lay your user's needs over the functional and business requirements. If it's not usable, then it's completely worthless to everyone involved. Address the user first and then work in the other requirements. Perhaps the hardest part is finding this common ground between the business needs and user accessibility, so do not be deterred.

Now, you may be asking "Why are we creating wireframes and mockups if we have already created a prototype?" That is a great question! Actually, what we want to accomplish at this stage (as opposed to the previous one) is a major commitment to the look and feel. The prototype should be created with the thought of being able to throw it away at a moment's notice. It is just there for you to have something to put in your hands, so to speak. This is the stage where we want to define the look and feel, make critical design decisions (like color scheme, layout, branding, etc.), and get ready to create the design document. Also, as a designer it can be useful to see your prototype in order to generate ideas faster (i.e., some of the thinking about how everything works together has been done) and it can provide a nice starting point from which to work from. Just don't allow yourself to become married to your own work at this point: be open to change and different ideas.

Now, that's why we create the mockups. The wireframes should simply follow the mockups, and here's why we create them. We are going to be drafting a design document in the next section. As you will see, one of the aims of this document is to allow all stakeholders in the project to see a kind of "Sneak Preview" of the upcoming functional spec. The best way to elicit useful and impartial feedback is to leave as much detail out as possible. Ideally, what you want to do is convey the application and the functionality without bogging the conversations down in discussions of color or branding. This is best done with wireframes. Wireframes are all about getting the right kind of feedback at the right time. So in order to gain useful comments about the layout, navigation, and functionality of the application, we want to strip out as much of the design and visual design as possible.

If it seems confusing to you to have a designer working on the look and feel while you are putting together a design document with only the wireframes, it may be useful to view this process as a spiral, or iterative, development cycle and not a linear, or waterfall, methodology. It has been my experience that I work faster and end up with a better product when I follow this sort of organic approach: letting several facets of the application design process run in parallel and gradually allow the application to sort of rise up on its own through the requirements gathering process. The key for all of this working is having a good team and keeping the communication open.

*****"Key pages" can be considered any page a user must interface with when using the application. Use your current table of contents to identify these and then remain flexible, as you'll be going through a lot of changing around.

Create A Design Document

As stated in the previous section, a design document serves to collect everything we know about the application at this point into a kind of pre-functional spec document and allow people the opportunity to review it and give their feedback. You will want to use this document as a tool for building consensus and, simultaneously, managing the expectations of people about what is coming. The key thing about design documents that you want to keep in mind is that they say very little (if anything) about the visual appearance of the application (aside from layout), and they say very little about the specifics of the user experience. If our models from Step 2 are at the 50,000 foot view and our finished spec will be at ground level, then this is somewhere in the neighborhood of 10,000 feet. We want to slowly bring the application into focus, reveal some details about the layout and overall structure of the site, and broadly highlight what functionality will be available to the user to do what. Beyond that, you are risking bogging down the project with frivolous discussions that do not matter at this point.

- **View an example design document at**
  http://www.mojofat.com/tutorial/designdoc.pdf

Write The Spec

Certainly, this is the most daunting task of the whole process. You've asked the questions, done the preliminary work, made a few mockups, and now it's just you with a blank Word document (or Framemaker, an XML editor, or whatever your preference may be) and a head full of ideas. Hopefully though, it's not totally blank. Throughout the first five steps, you've ideally been taking notes, developed a working outline, and maybe even started writing a few sections as they come up. Although each writer will invariably have their own style, I would like to pass on a few general tips that you may find useful in structuring your document:

Cover everything

By this, I mean make sure you've written about every single interaction point that exists in the application. Don't leave anything to guess or assume. It's quite likely that you'll come to a point where you believe you have included everything, only to find you have left out something very obvious. Every link, text field, button, etc. should have a detailed description of what it does and how the user experiences it.

Use lots of screen shots

It's usually helpful to slice up your mockups and insert screenshots of the individual elements you may be writing about. And always include a screen shot of the entire page at the end of its section.

Write concisely, correctly, and consistently

An economy of words will be appreciated by those who have to read your document and make sense of it all. Additionally, try and break things out into logical components or steps as much as possible. By writing correctly and consistently, I mean make sure you're using proper grammar and consistent terms (e.g., if you use "drop down menu" in one part of your document, don't start using "pull down menu" in another part to mean the same thing).

Use the tools and format most comfortable for you

As a spec writer, you still have quite a few decisions make. What tools do you use? What format? What sort of style should you adopt? Etc. Also, you need to take into account how your documents will be used: will they need to be online, will they frequently be merged with other documents (e.g., large technical libraries), do they need to be in a particular format or follow a certain style? Really, there is no single correct path for all of this, and it depends on you: the writer. I have frequently used Word, PhotoShop, and Visio to create my specs. This setup has its pros and cons, but I have also found myself in a position where I needed to adopt the system of my employer…or one of their clients, etc. This has led me down the DocBook/XML path, the PageMaker path, and the FrameMaker path. Like I said, it really just depends on a number of variables; the most important being what you prefer and how your documents will be used.

For more on these tools, be sure to check out the "**Resources**" section at the end of this tutorial.

Edit And Rewrite

Rewriting and editing are, literally, half the battle. When setting the timeline for your completion of the spec, always try to add a week more than you anticipate needing. In my experience, something always comes up during the editing process: I've forgotten something, someone now wants to add a new element, details need to be filled in, etc. Since so many people from so many different areas of expertise will be reviewing this document, it is important that it is perfect: no spelling errors, no grammatical errors, no logic errors, and no mock up errors either. To make sure I catch everything, I typically end up printing out my spec many times and editing with a pen. Some basic things to do during the editing process:

Check your table of contents

This is an obvious one, but make sure your table of contents correlates exactly with what's in your document.

Edit from beginning to end at least three times after you think it is done

Three times seems to usually do the trick...sometimes more is required, but never less. The first pass usually involves a lot of rewriting and renumbering, while the second pass is typically much lighter and may catch some of the obvious things you missed the first time. By the third time, you should be merely polishing and fixing very small details.

Have someone proofread for you

This is optional, but I always find that it helps a great deal to have someone else read it over for you (especially if that person has a strong command of the written word).

Review

Whether you're writing a spec for your company or for a client, you're going to have a series of people reviewing your spec to make sure they understand what's going to be built. Expect changes and be prepared to give reasons for why everything is written as it is. By the time you take your spec to any clients or colleagues, it should be airtight. Part of the review process is working with your client or colleagues to come up with the best solution. If they feel the application will not meet their needs as specified, then find out why and offer solutions as to what can be done to meet their needs. After the review, take your notes back to the computer and make your changes while they're still fresh in your mind. Be sure to change your mockups and all related text, double check to make sure your changes don't affect other functionality (if it does, remedy it), and then double check your table of contents.

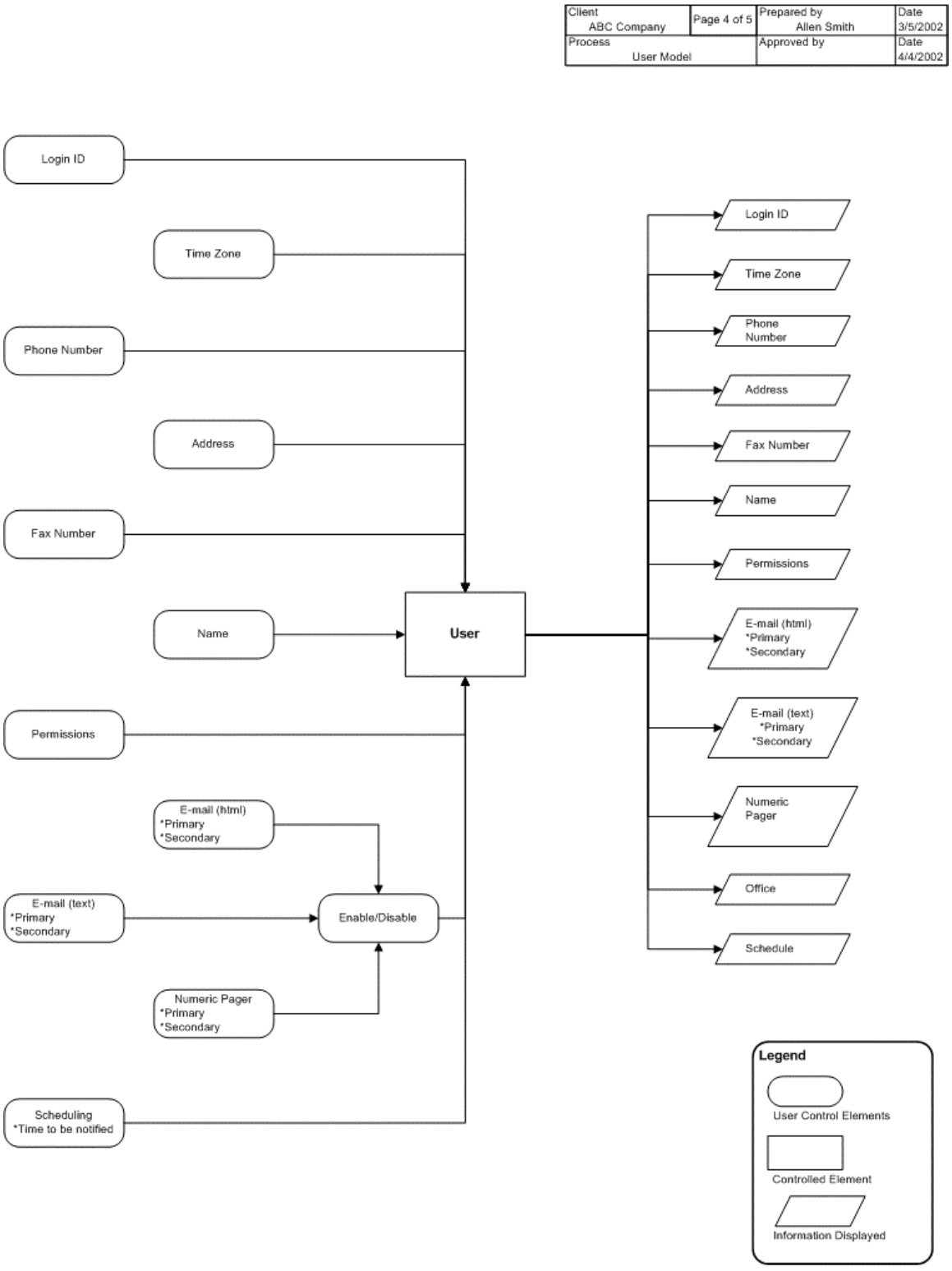
Prepare the functional spec for another review. Have everybody signoff on it once a consensus has been made.

Conclusion

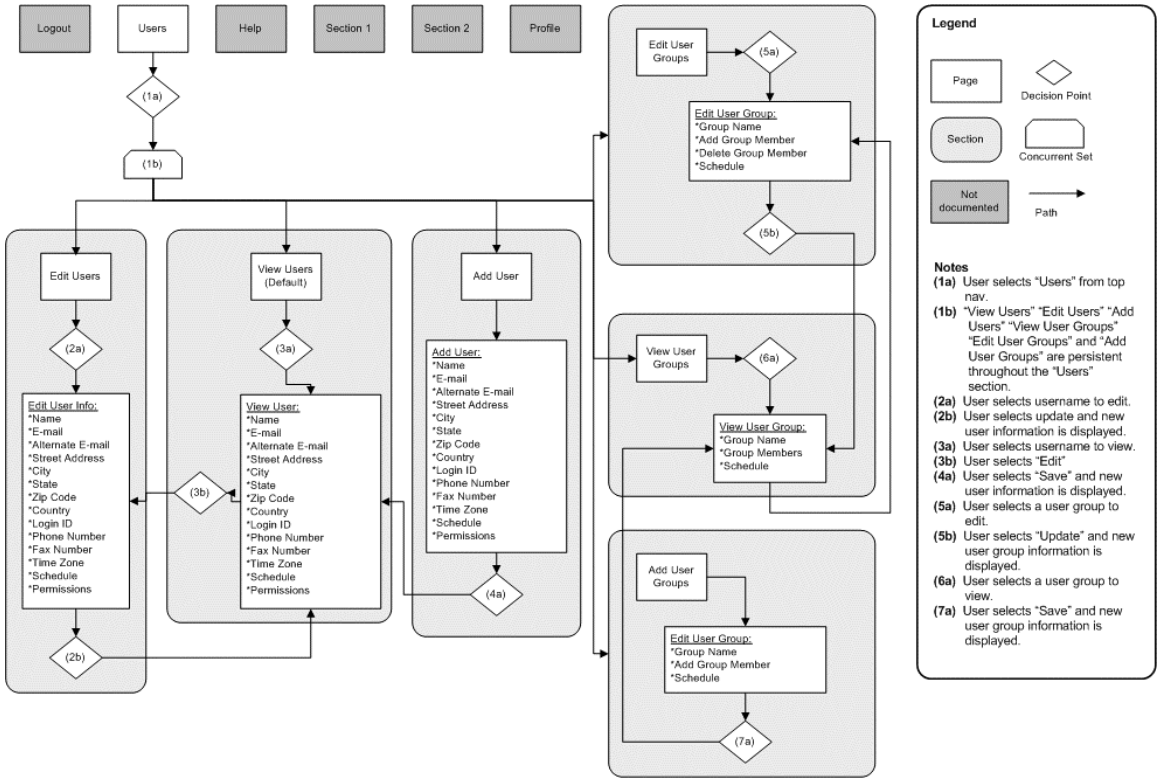
Now that you have a finished and signed off spec, you're ready to begin development. Be prepared to answer questions as people begin developing the application from your spec. By going through this process and writing everything down, you'll see a decrease in development time and a more efficient application development process. You may even see better applications being built and happier customers!

Appendix

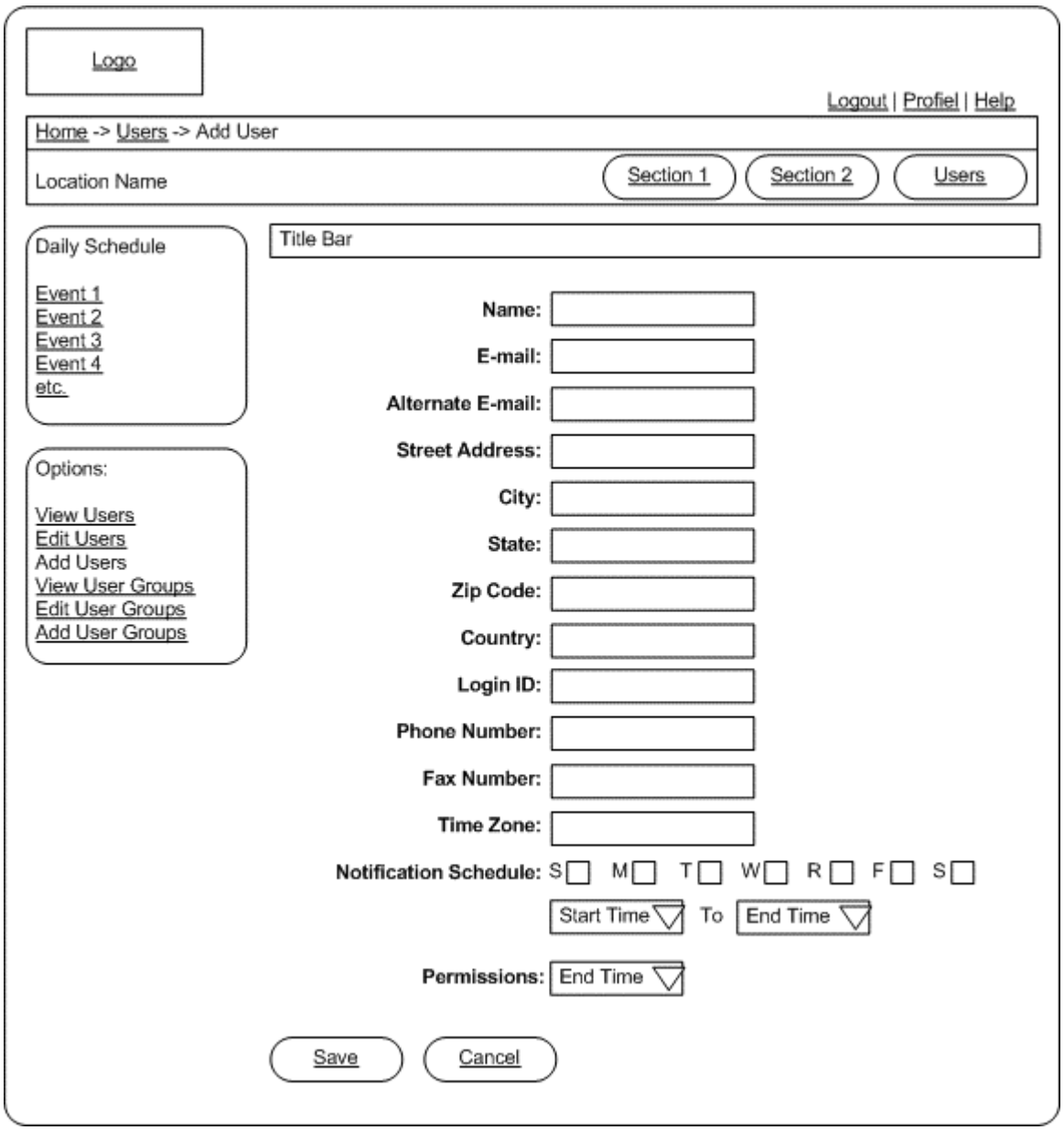
Designer's Model



Flow Diagram



Wireframe


Logo
Logout | Profiel | Help
Home -> Users -> Add User
Location Name
Section 1
Section 2
Users
Daily Schedule
Event 1
Event 2
Event 3
Event 4
etc.
Options:
View Users
Edit Users
Add Users
View User Groups
Edit User Groups
Add User Groups
Title Bar
Name:
E-mail:
Alternate E-mail:
Street Address:
City:
State:
Zip Code:
Country:
Login ID:
Phone Number:
Fax Number:
Time Zone:
Notification Schedule: S M T W R F S
Start Time
To
End Time
Permissions:
End Time
Save
Cancel

Resources

Websites

Although there are a scant few sites out there that deal with Functional Specification Documents directly, there are many resources you can use for various aspects of spec writing. The following sites are just a handful of the really good informational sources out there relating to (in one way or another) the discipline of creating Functional Specs: usability, visual design, architecting information, web design, etc.

Sample Functional Spec
View a chapter from a spec I wrote (*http://www.mojofat.com/tutorial/MBSpecCh5.pdf*).

Design Document Sample
View a sample design document I created for the online tutorial (*http://www.mojofat.com/tutorial/designdoc.pdf*).

xblog
Visual thinking weblog (*http://www.xplane.com/xblog/*).

elegant hack
Information architecture reading list (*http://eleganthack.com/reading/index.html*).

peterme
Interface design reading list (*http://www.peterme.com/Interface*).

metagrrl
Design process in online Environments (*http://www.metagrrrl.com/design/proposal.htm*).

Usable Web
Links to web usability articles (*http://usableweb.com/topics/000505-0-0.html*).

Webword
Usability (*http://www.webword.com/*).

Use It
Usability (*http://www.useit.com/*).

Joel on Software
Good look at why functional specs should be written (*http://joel.editthispage.com/stories/storyReader$212*).

Webmonkey IA Tutorial
Information architecture. (*http://hotwired.lycos.com/webmonkey/design/site_building/tutorials/tutorial1.html*).

Photoshop Workshop
Photoshop tutorials (*http://rainworld.com/psworkshop/*).

Boxes and Arrows
The definitive source for the complex task of bringing architecture and design to the digital landscape (*http://www.boxesandarrows.com/*).

STC Reference Documents
This collection of reference documents and templates is housed by the Society for Technical Communication (*http://www.stcsig.org/mgt/reference.htm*).

TECHWR-L Document Types
Similar to the above STC, here you can find many tools and examples for the functional spec writer (although not always aimed squarely for functional specs) (*http://www.raycomm.com/techwhirl/magazine/writing/doctypeshomepage.html*).

Software Usability Research Lab
You can find some very good articles about web usability and designing a strong web experience (*http://wsupsy.psy.twsu.edu/optimalweb/*).

Seven Pitfalls to Avoid in IA
Interesting and relevant article highlighting the common things to avoid when structuring your information. (*http://www.internetworld.com/magazine.php?inc=121500/12.15.00feature3long.html*).

How Non-Programmers Use Documentation
This article points to the ways non-programmer type people use documentation and how you can use that information. A useful perspective since at least some of

your readers will undoubtedly be non-programmers (*http://advogato.org/article/374.html*).

Strategies of Influence in Interaction Design

Funny thing is, is that the scenario described in this article pretty much happened to me as well. Read this and learn how to manage the other part of your spec: the office politics (*http://www.uiweb.com/issues/issue18.htm*).

Representations and Perceived Information Architecture

This article discusses two key ideas. First, it briefly outlines four ways to represent the same information. Second, it provides a high level overview of Perceived Information Architecture (*http://webword.com/moving/representations.html*).

Downloadable Visio Shape Libraries

jjg.net has a fine downloadable shape library for Visio. I suggest complimenting this stencil with your own select shapes from Visio's libraries (*http://www.jjg.net/ia/visvocab/#download*).

Tools

The following the list of tools I currently know about that are currently used by spec writers of all stripes. If I've missed anything, be sure to let me know (*al@mojofat.com*)!

DocBook

DocBook is a widely used DTD for creating technical documents in XML and SGML. Be prepared to learn about open-source tools if you go this direction (that's either a good or bad thing, depending on who you are).

OASIS

OASIS maintains the most current version of DocBook, and also has a lot of other useful information for the DocBook user.

XML Pro

If you're using DocBook then you'll probably want some sort of XML editor (of course, you can always use your own favorite text editor if you wish). XML Pro is a pretty good one (I haven't used it extensively though).

FrameMaker

Depending on what you're working on, FrameMaker might be an overkill. Typically, my functional specs are no more than 100 pages. FrameMaker is more designed for 500+ page documents, but again: it's a personal preference.

PageMaker

PageMaker can also be used by your average spec writer. The advantage here is the ability to manage multiple documents and assemble them into books.

PhotoShop

I don't know what I would do without PhotoShop. For me, this is a must-have.

Word

Of course, there's always the standby: Word. This is the primary tool I use to write, but I am trying to move away from it. My chief complaint against word is that I find it a bit inefficient when producing large documents...especially those with pictures and diagrams.

Visio

Visio is another tool that I find very, very useful. This is a program you use to create diagrams, flowcharts, models, etc.

Acrobat

This is what you use to convert your specs into .PDFs. Having your spec in a .pdf format is useful because it is a great choice for sharing and protecting your documents.

Google

Ok, this may not fall in line exactly with the others listed but Google is perhaps the one tool I use the most out of all of them! It's an absolute necessity for researching your application.