What is a Functional Specification?

- They are the blueprint for how you want a particular software product 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.

Functional Specification Vs Design

- A functional specification states what the proposed system is to do.
- A design is how the system is to be constructed to meet the functional specification.
- In writing a functional specification, some consideration of design issues must take place, to ensure a realistic system is specified.

Functional Vs Technical Spec’s

- A functional specification describes how a product will work entirely from the user's perspective.
  - It doesn’t care how the thing is implemented.
  - It talks about features.
  - It specifies screens, menus, dialogs, and so on.
- A technical specification describes the internal implementation of the program.
  - It talks about data structures, relational database models, choice of programming languages and tools, algorithms, etc.

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 (supervisor in your case), they are building what the client is expecting.
- There should be nothing left to guess or interpret when the spec is completed.

Why won’t people write specs?

- I think it’s because so many people don’t like to write.
- People claim that it’s because they’re saving time by skipping the spec-writing phase.
- They consider spec-writing a luxury reserved for NASA or people who work for giant, established companies.
- Failing to write a spec is the single biggest unnecessary risk you take in a software project.
  - It’s as stupid as setting off to cross the Mojave Desert with just the clothes on your back, hoping to “wing it.”
- Programmers and software engineers who dive into code without writing a spec tend to think they’re cool gunslingers, shooting from the hip.
  - They’re not.
  - They are terribly unproductive.
  - They write bad code and produce shoddy software, and they threaten their projects by taking giant, completely uncalled-for risks.
**Functional Spec = Decisions**

- In too many programming organizations, every time there's a design debate, nobody ever manages to make a decision, usually for political reasons.
- So the programmers only work on uncontroversial stuff. As time goes on, all the hard decisions are pushed to the end.
- These are the most likely projects to fail.
- Writing a spec is a great way to nail down all those irritating design decisions, large and small, that get covered up if you don't have a spec.
- When you force yourself to write a good, complete spec (and I'll talk a lot more about that soon), you should notice all the things that need to be included and you either fix them or at least you mark them with a big red flag.

**Who writes a Functional Spec?**

- For CA4 project... YOU
- Ideally:
  - 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 the problem to be addressed, 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 you may navigate back and forth through the information.
  - Not only do you need to map this world out usually, but you also have to work on a great deal of detailing the entire world, 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.

**Do all projects need one?**

- Yes for any non-trivial project.
- A non-trivial = more than about 1 week of coding or more than 1 programmer.
- If you don’t have a spec, you will always spend more time and create lower quality code.
- Here’s why:
  - The most important function of a spec is to design the program.
  - Even if you are working on code all by yourself and you write a spec solely for your own benefit, the act of writing the spec - describing how the program works in detail - will force you to actually design the program.

**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 checklist of general questions you should be asking:
  - Depending on what the project is, you’ll probably come up with more specific questions.

**A suggested high-level approach**

- Define the application.
- Develop models.
- Define information flows.
- Write the spec.
- Review and edit.

**Define The Application... Steps**

1. What is the application supposed to be?
   - Make sure you have a strong grasp on what the product is before you start anything.
2. What is the application supposed to do?
   - Once you know what the product is supposed to be, you should start dissecting what it is supposed to do.
   - Identify the main features of the product based on its 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.
   - 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.
   - A good way to do this is to create a matrix of all the functions to want the application to perform and then rank them based on priority.
3. Who is going to be using this application?
   - Find out what the audience is.
   - Create scenarios and develop personas.
4. Is there a precedent for this application?
   - If what you’re developing is similar to other products out there.
   - Perform a competitive analysis... get idea of the competition.
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.
- Models are an aid to explain complex ideas.
- As the functional spec writer, you are responsible for creating information models to help convey the concepts of the application you are developing.
- Use case are often useful at this stage.

Define The Information Flow
- To start this process, try and make an outline of everything you currently know.
- A lot of this should be easy after creating your models for the application.
- Maybe 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 complex set.
- Anything you develop now will change over time, but ultimately you will have a thorough flowchart of the application included with your spec.
- Mind is a good tool for developing technical diagrams and flow charts such as these.

Edit And Rewrite
- Rewriting and editing are, literally, half the battle.
- When setting the timeline for your completion of the spec, always try to allow more time than you need.
- Something always comes up during the editing process.
- Some basic things to do during the editing process:
  - Check your table of contents.
  - Try to do a general edit so you make sure your entire document contains a verb and a subject.
  - Still from beginning to end at least three times after you think it is done.
  - Have someone proofread for you.
  - Use proper grammar and consistent terms.

Write the Spec
- General tips that you may find useful in structuring your document:
  - Cover everything.
  - Don’t leave anything to guess or assume.
  - Use lots of screen shots.
  - Try and break things out into logical components or steps as much as possible.
  - Use proper grammar and consistent terms.
  - How are your documents used? Will they need to be merged with other documents, do they need to be in a particular format or follow a certain style?

What goes into it? (1)
- Table of contents
  - Indicated for all sections / headings and page numbers.
- Introduction
  - Overview:
  - Provides a brief overview of the system / product to be developed.
  - It should include a description of the need for the system, briefly describe its functions and explain how it will work with other systems (if appropriate).
- Business Context
  - Provides an overview of the business context in which the system / product will be deployed.
  - May not be applicable to all projects.
- Glossary
  - Define and technical terms used in this document.
  - Only include those with which the reader may not be familiar.
**What goes into it?**

(2)

- **General Description**
  - Product / System Functions
    - Describes the general functionality of the system / product.
  - User Characteristics and Objectives
    - Describes the features of the user community.
    - Explains the objectives and requirements for the system from the user's perspective.
    - May include a "wish list" of desirable characteristics, along with more feasible solutions.
  - Operational Scenarios
    - This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations.
  - Constraints
    - Speed, industry protocols, hardware platforms, etc.

---

**What goes into it?**

(3)

- **Functional Requirements**
  - This section lists the functional requirements in ranked order.
  - Functional requirements describe the possible effects of a software system, in other words, what the system must accomplish.
  - Non-functional requirements (such as interface requirements, performance requirements, or reliability requirements) describe how the system accomplishes its functional requirements.
  - Each functional requirement could be specified in a format similar to the following:
    - **Description** - Full description of the requirement.
    - **Criticality** - How essential it is to the overall system.
    - **Technical Issues** - Describes any design or implementation issues involved in satisfying this requirement.
    - **Dependencies with other requirements** - Describes interactions with other requirements.

---

**What goes into it?**

(4)

- **System Architecture**
  - Describes a high-level overview of the anticipated system architecture showing the distribution of functions across (potential) system modules.
  - Architectural components that are reused or 3rd party should be highlighted.

- **High-Level Design**
  - This section should set out the high-level design of the system.
  - It should include one or more system models showing the relationship between system components and the systems and its environment.
  - These might be object models, DFD, UML, etc.

---

**What goes into it?**

(5)

- **Preliminary Schedule**
  - This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates.
  - The plan also includes information on hardware, software, and systems resource requirements.
  - The project plan should be accompanied by one or more PERT or GANTT charts.

- **Appendices**
  - Specifies other useful information for understanding the requirements.