Developing Your System Requirements
Checklist

Instructions:
What truly defines your system is the functions it must perform in order to successfully accomplish its use case. If you're able to formalize these functional requirements, you have created a technical definition of what any valid solution to your problem must do.

The ten steps in this checklist will help you analyze the use cases and formally determine functional requirements that are required of any valid solution to the challenge you are trying to solve.

Steps	Is Complete When	✓
Before You Begin: Define your stakeholders and your system's context and interfaces, and prepare a list of use cases to explore. If you are working with a specific client, you will also want to define your client's deliverables.	You have defined your stakeholders, you understand your system's context and interfaces, and you have developed a list of use cases to explore further.	
Step 1: Rate your use cases to the best of your current knowledge as to which ones are the most important.	You have prioritized your use cases in some way that identifies which use cases you currently consider to be the higher priority.	
Step 2: Select one of your higher priority use cases to explore further by starting to create a Use Case Behavioral Diagram (UCBD). A UCBD is a way to describe, step-by-step, what your system must *functionally* do and what other objects/users/etc., are doing during that use case.	You have copied the template or created a similar UCBD setup and have labeled it with your use case name.	

Steps	Is Complete When	✓
Step 3: To begin creating your UCBD, determine who/what are the main *actors* involved in this use case. Label a separate column for each actor, as well as a column for your system. Traditionally, the main actor(s) are placed in columns to the left of your system's column. Other actors' columns are thereby placed to the right of your system's column.	You have created a separate column for each actor and one for your system according to the guidelines.	
Step 3a: Jot down notes as you go through the UCBD process to help provide the reader insights on any possible misconceptions. The bottom of the UCBD is reserved to list notes for the reader. Notes are often written informally, but each one is numbered so it can be easily referenced later on.	You have added a Notes section to your UCBD an included any notes you think the reader may need. (You can return back to this step as often as needed throughout your UCBD process.)	
Step 3b: Be sure to create one—and only one—column for your system. Do not split your system into subsystems at this stage.	You have only one column for your system in your UCBD. If necessary, you have accepted a placeholder concept to start your UCBD process.	
Step 4: Establish what are the starting conditions for your use case. You may have a starting condition, or even multiple starting conditions, for each actor and your system.	You have established the starting condition(s) for your use case and listed it in your UCBD.	

Steps	Is Complete When	✓
Step 5: Establish what are the ending conditions for your use case. This can be how your use case scenario should end, placing it in a fairly stable state, or it could be in transition, as perhaps the starting conditions for another use case.	You have established the ending condition(s) of your use case.	
Step 6: Explain what must functionally occur from your use case's starting condition to its ending condition. Often, but not always, the use case begins with some kind of a trigger action that will cause your system to do something. Actions done by any of the actors are typically written in a slightly less formal way, simply stating what is occurring. What your system should do, however, should be written very formally as *functional requirements*.	You have a series of actor statements and system requirements that describe what must occur and the functionality your system must provide/perform in order to advance from the use case's starting condition to the use case's ending condition.	
Step 7: As you review your UCBD, make sure that you've written your requirements "functionally, not structurally." This will become natural in time, but when you are starting out use resources to aid the writing of functional requirements. Functionally written requirements help ensure you are not artificially constrained as to what your system must structurally be. Instead, you are open to creatively think about possible solutions that could meet your challenge's functional needs.	You have checked all of your requirements to make sure they are written functionally, not structurally.	

Steps	Is Complete When	✓
Step 8: A cornerstone of professional design is well-written requirements. Your requirements should be: • Written as "shall" statements • Correct: what you're saying is accurate • Clear and precise: one idea per requirement. For the word "and" or similar conjunctions it's considered better to split the requirement into two. • Unambiguous: only one way to interpret • Objective: non-opinionated • Verifiable: there is some measurable way you could say this requirement is met • Consistent: does not contradict another requirement	You are confident that your requirements are written in a professional manner.	
Step 9: Review your use cases for missed functionality. It's rare that even the most experienced professional designer will think of all of the key functions or delve far enough on their first try, so it's quite common that your first pass at a UCBD might be pretty short and rather high level. Go over it again, and maybe again, asking yourself questions like: • "If the system has to be able to do this, what else must it do?" • "Are there other functions that are occurring at the same time?" • "If I asked a contractor to create something that just performed the functions I wrote and nothing else, would I be confident that what I got	You have reviewed your use cases and requirements using the questions provided here and the delving tips suggested previously, refining or adding any additional requirements to your UCBD.	

Steps	Is Complete When	✓
back would be able to meet all of needs associated with this use case?" These are tough questions, but continue with them until you feel you can functionally define what any system has to be in order to meet the needs of this use case. Taken together, anything that meets the collection of requirements across your use cases is a valid solution. It is only through creating more functional requirements that what the system should be begins to take shape.		
Step 10: Repeat Steps 2–9 for your remaining high priority use cases and some of your medium, or even low, priority uses cases. You most likely do not have the time to create UCBDs for all of your use cases, so when trying to choose which lower priority use cases to explore, try to select the ones that you think will involve different kinds of functionality and set different kinds of requirements for your system than those you have already declared.	You have a list of functional requirements that define the functionality that your system must achieve.	