330 Writing #3 – Project Specification Document

Your group's goal is to flesh out exactly what functionality needs to be implemented in future phases of your project. You're starting with a project description that gives the flavor of the desired product, but which is vague on some issues and silent on others. As it stands, there is considerable ambiguity about exactly what a user will be able to do with the system. You may have read it and formed your own opinions, but if you started to design the system now you would quickly realize that many of the assumptions you had made would not be the same assumptions that your fellow team members made, or that a stakeholder might have made. At the end of this phase, the requirements should be spelled out in sufficient detail such that there is very little left unsaid about *what* the system will do and how you will accomplish it.

Your document should be written for the stakeholders of the system. A wide audience, including developers, clients (people requesting the system), and end users, may read it. Try to avoid unnecessary technical jargon.

You should provide an overview of the document and your system. Describe in detail the concept of the system that you are building. Document the functionality that the system will have by including use case descriptions and diagrams. Document your system design.

Very important notes

I. Describing a feature is a heck of a lot quicker than designing it, which in turn is a heck of a lot quicker than coding it and testing it. Therefore, you will be able to describe far more features than you will actually have time to implement in this course.

For this reason, please understand that **you describing a feature in your use case model does <u>not</u> obligate you to actually implement that feature in code later on.**

Each team member should author at least four use cases. If you're planning on building a pretty bare bones Twitter system, your final application might not even *have* twelve use cases. That's okay. *Imagine* sufficient functionality in this phase to have a good twelve or more. If your final product doesn't build all that, it's okay.

II. Note that as a team, you should definitely review each other's work and give feedback. This is the only way that a truly quality result will emerge.

III. Remember the goal. The goal is not to do a bunch of busy work, but to elucidate the system behavior so that your team is all on the same page when you begin design. To be honest, the deliverables themselves are just the icing on the cake - the cake itself is the process of exploring and brainstorming and negotiating so that you emerge from this phase with a thorough understanding of what it is you want to build. The deliverables are simply the tangible record of that understanding.

Stuff always changes, and as you begin design and implementation in later phases, you will inevitably change your mind about things and encounter issues that you didn't think of before. That's okay. That's part of software development. You can go back and refine your use cases in later phases as you discover these issues and confront the reality of change.

Think ahead. The use cases you write in phase 1 will have a profound effect on what happens later. You want to resolve *any* ambiguity about *what* the system is going to do, so that you have a joint understanding moving into the implementation phase.

IV. In addition to including diagrams, explain everything in text form.

V. Don't forget to number and label all of the figures that you include.

TEMPLATE

Title Page with Index

1. Introduction

This section should explain to the reader why you are writing this document and how it is organized. It will also give a "big picture" overview of your system.

1.1. Document Purpose

Explain the purpose of this specification document.

Make sure that you identify the intended audience for the document.

1.2. System Scope

Briefly describe the system that you're building. Provide the motivation for this particular system. Include a very high level overview. You'll explain the system in more detail in the next section.

1.3. Overview of the remainder of the document

Explain how your document is organized. Write a sentence or two summarizing each large section so that if the reader wanted to know where to find some information about the system, they'd have a good idea of where to look within this document.

2. Overall Description

This section should discuss background information about your system including a definition of client/s, definition of users, and detailed description of the included functionality. This information should be broken into subsections. One example breakdown is shown below.

2.1. Client characteristics

Who is your client? Why is this system important to him/her/them? This is fictional and can be made-up.

2.2. User characteristics

Who are the intended users of the system? Describe them.

2.3. Product functions

Here's where you should describe the software in more detail than you did in section 1.2. What type of functionality will be included?

Describe the features in detail. Document the system functionality with a set of use case write-ups and a use case diagram.

3. System Specification

Explain your system design. What data will be stored in each class? What operations will each class include? What type of relationships will exist between classes in your system? How will the various classes interact? In addition to your text explanations, include a class diagram for your system.

4. Assumptions

List any assumptions that you have made about the system. You should have a brief introduction statement/paragraph in this section. The actual assumptions should be communicated as a bulleted list.

Turn in: One document for your group.

This document should be priority number one this week. As a group, you need to have a consistent vision for the system, how it will be used, what it includes, what it omits, and how it is organized. It's best if you suspend implementation for just a bit. Get all the specifics worked out. (You may still be working on some research about useful things included in the Java API and setting up your version control system, but don't get heavily involved in the implementation just now.)