Software Requirements Specification

Due:

Overview

The goal of this assignment is to establish a solid definition of your project from which to base a design and implementation.

Your first deliverable is a set of requirements documents (sometimes called “Software Requirements Specification” or SRS). These describe the goals of your project and how users will interact with it (the high-level UI design). You will also document your plans for completing the project.

Note that this document will be a living document. You will be asked to provide updates to it at periodic points in the development cycle. Furthermore, parts of it are an update to your proposal.

Deliverables

You will submit 4 brief documents:

1. **A product description**
2. **UI diagrams**
3. **Use cases**
4. **A process description**

The product and process descriptions together should be 4-6 pages total. There is no page limit on the UI diagrams and use cases, but don’t overwhelm the TAs. All documents must be in PDF format.

Only one person from your group should submit your documents via the github. Choose a short name for your team (based on your project name). Each document should be named `TeamName_DocName.pdf`, where <*TeamName* > is the name of your team and `DocName` reflects the documents contents—for example `PacMan_UseCases.pdf`. You may receive a deduction if you turn in clumsily named files. Make sure that your project’s name and all group members’ names appear clearly atop each document.

External requirements

We require your product to be as usable as possible. In particular:

- The product must be robust against errors that can reasonably be expected to occur, such as invalid user input, lost network connections, etc.

- The product must be installable by a user, or if the product is a web-based service, the server must have a public URL that others can use to access it. If the product is a stand-alone application, you will be expected to provide a reasonable means for others to easily download, install, and run it. Your instructions should assume that your target user is using a UW CSE Linux VM.
- The software (all parts, including clients and servers) should be buildable from source by others. If your project is a web-based server, you will need to provide instructions for someone else setting up a new server. Your system should be well-documented to enable new developers to make enhancements.

The scope of the project must match the resources assigned.

Product description

You can think of this as an enhanced and refined version of your product proposal.

What is your product? Who is the target audience you expect to use the product? What problem does it solve?

What alternatives exist, and what are their strengths and weaknesses? How will your system be different, from the user’s point of view? Be specific.

What are its major features? Include at least 4 major features you will provide, along with at least 2 “stretch” features you hope to implement but that could slip to version 2.0 if necessary.

What are its non-functional requirements?

What external documentation will you provide that will enable users to understand and use your product? This could take the form of help files, a written manual, integrated help text throughout the UI, etc.

UI diagrams

What will the UI look like? Submit **at least two diagrams** containing rough sketches of your product’s user interface, when executing your use cases. By “diagram” we mean a depiction of a complete major screen, web page, window, etc. of your system. You should show at least all screens that a user would see when executing your use cases. From looking at these UI diagrams, the customer should be able to clearly see how your product will be able to successfully complete each use case you are submitting.

These diagrams should depict the major UI used to complete the use-cases you submit. For example, if one of your use cases is to Purchase Tulip Bulbs, you should draw the initial UI that is presented when the user wishes to purchase a tulip bulb, along with any other major windows, messages, etc. that appear as the user navigates through this use case. If a window leads to a dialog box, drop-down box, etc., include it as a sub-diagram.

Your diagrams do not need to be beautiful to get full credit, but they should be clear and legible. They should reflect forethought about what information needs to be shown and how the user will use the software. Part of your grade comes from choosing appropriate UI elements to effectively accomplish the desired task.

To help the grader/customer understand how to “use” your prototype, it would be helpful if you labeled which items and sheets go together, for example, with numbers or names. For sub-diagrams such as pop-up windows, drop-down boxes, etc., it is helpful to put a corresponding number/name in the place that sub-diagram appears. If there is anything non-obvious about how your UI is used, you may include brief written instructions about how the user is expected to walk through the UI.

Your group will not be “locked in” to using the exact UI represented by these diagrams on your actual final project. The purpose of a prototype is to try out user interface ideas, see how they work, and revise/modify the UI as needed. By being concrete now, you will make discoveries early and at relatively low cost.

Use cases

Submit **at least three use cases** for scenarios you think are the most important to your product. Make a brief, informal argument that your set of use cases covers the important scenarios (perhaps by referring back to the product description).

The format of the use cases is up to you, but these aspects must be clear: actors, preconditions, triggers (what starts the use case), minimal/success guarantees (end condition), the list of steps to the success scenario, failure end conditions, and extensions/variations of the scenario.

Also discuss failure-handling remedies as appropriate (or state that you do not have one and what you specific steps you will take to generate them). It is impossible to think of every possible use case or failure mode ahead of time. But you should list a reasonable set of extensions and remedies if reasonable ones exist.

Process

Software Toolset: What programming languages, data sources, version control, bug tracking, and other tools will you use? What, if any, software components will you attempt to use “off the shelf” versus implementing them from scratch? Explain why you chose these tools and languages, as well as why they are suitable for use on this project.

Group Dynamics: We require that you choose a single person to serve as your Project Manager (PM). Who will be your project manager? What will be the other members’ roles? Will everyone share in the development, or will you have designated designers, testers, etc.? Why have you chosen these roles? Will the roles differ for different parts of the project? (This may require you to think a little bit about how your system will be implemented.) If a disagreement arises, how will it be resolved? Be specific. Also justify your decisions by briefly explaining why you have chosen this structure.

Schedule / Timeline: Provide a rough schedule for each member or sub-group within your team. For example, how long do you believe your developers will spend working on each major feature listed in your product description? Who will work on the design, and how much time do you expect it will take? For each feature, when will it be complete (give a milestone, such as by the beta release)? Provide reasonable guesses as much as possible. You will not be graded on the accuracy of these predictions. You will, however, be graded on their reasonableness, completeness, and justifications.

Risk Summary: What are the major risks (**at least three** of them) regarding completing your project? What are you most worried about, and why are these the most serious risks? Describe specific experiments that you will perform to gather information that will reduce the risk. Also describe what you will do if you are unable to overcome the problems—for example, if you cannot get an external component to work, or if you fall behind schedule. This might include feature cuts, but you may no use “feature cut” as all of your mitigation strategies.

As a special case of risk reduction, describe at what point(s) in your process feedback from an external user evaluation will be most useful, and how you will get that feedback.

Hints

Part of your grade will come from the plausibility, thoughtfulness, and level of detail of your work. For example, when listing software tools, list a plausible set of tools for all aspects of the project and defend your choices.

In use cases, sloppy or incomplete work often leads to deductions. We want to see that you have given due thought by choosing substantial use cases that are important to the core functionality of your product. You should also list a reasonable set of steps in the various scenarios that can occur in these use cases. Take care not to omit important steps or details. Make sure that the state of the system at the end of any path through the use case matches the guarantees that the use case claimed would occur.

Your documents must be clear and professional. This means they should be concisely written, with proper spelling and grammar, clear wording, and formatted with a enough organization to present your ideas clearly to the reader.