Computer Science Capstone Design
Assignment: Software Test Plan

Introduction

A test plan outlines activities that are aimed at ensuring that your project's implementation exhibits the necessary functional and non-functional characteristics. In this document, you are asked to describe how you intend to ensure that the expectations presented in the requirements and design specification documents are met, via a well-planned software testing regime. If you feel your project should not involve one or more of these testing types, discuss the issue with your team mentor -- and be prepared to offer convincing rationale, since these testing types are widely prevalent and mostly applicable.

This assignment write-up assumes that you are familiar with the basics of testing and quality assurance through your software engineering and other course materials, so it is relatively "light" on the details. Your past course material should give you a good reference point for these details.

Typical Content Outline:

Cover Page

The customary cover page with your team's name and the names of your team's members. Include some indication of which revision of the document this is (for example, "Version 1" or "Version 2.3").

Table of Contents

The contents of the document, and the page number on which each section begins. Just reference major sections (not every subsection); the TOC should never have two entries that point to the same page.

Introduction (about 1 page)

Begin with a brief paragraph summarizing the motivations, nature and primary goals of your application. This just sets the stage so that we know *what* we're testing here. Follow this with a paragraph that introduces software testing (motivations, what is it?) generally. Next you'll get to the heart of your intro: A short summary that briefly describes your overall plans (what kinds of testing, how many/much), and summarizes activities related to each type of testing you will engage in. This is basically an outline for the discussion coming up in the core of the document in upcoming sections. Close your intro with a discussion of *why* your outlined testing plan looks the way you've presented it, the *rationale* for *this particular testing regime.* Base this discussion on the nature of your application…why have more intensive testing in one area than another for this app? How did the nature of app/users/context lead you to design this testing protocol? Don't forget your segue introducing the remaining sections at the end of this part!

Unit Testing (as needed, about 4-5 pages)

This section should discuss your plans for creating unit tests aimed at ensuring that key methods and/or procedures function correctly. Start with a brief intro explaining what unit testing is, what

it's goals are, and your detailed process/tools for doing the testing, i.e., what unit test libraries/tools you are using to streamline your testing activities and any test-related metrics you will be determining (test coverage, for example). Consult your software engineering course materials if you are foggy on details here! Then give a listing of what the "units" are that you will be testing for your project. If you are not conducting detailed unit testing for each method and/or procedure, begin by outlining your rationale for only focusing on a subset of your system.

Continue by presenting a detailed plan for testing your code: *For each unit of code you're testing*, specify the equivalence partitions and boundary values you've identified and present selected inputs from these partitions and boundary values. Also make sure you include cases with erroneous inputs addressing the robustness of your code.

Integration Testing (as needed, about 2-3 pages)

Integration testing is focused on the *interfaces between* major modules and components, and focuses on whether the interactions and data exchanges between modules take place correctly. At a very simple level, for example, unit testing may focus on whether a method call returns the correct result, while integration testing is focused on whether the data for parameters and return values is exchanged correctly. Even more simply, integration testing usually focuses on the "plumbing" of a system, i.e., if everything is wired together right. You should focus your integration testing on the "boundaries" between important modules of your system, such as those allowing access to databases, elements involved in the data exchange of a web-based interface, or network-based communication.

To lead your reader into this section, start with an overview of Integration Testing: what it is, what its goals are, and what your overall approach procedure was in determining what to test and how. Basically: you are giving an overview and rationale for your proposed effort. Then present a detailed plan for testing the integration of major modules in your code: For each integration point, specify what test harnesses you will use and how you will verify that modules integrate correctly, ensuring that interactions take place correctly and all contract assumptions (such as data needed to invoke a function and data required to be returned) are respected.

Usability Testing (as needed, about 2-3 pages)

Usability testing is focused on the interactions between the software system and the end user, and is intended to ensure that users can effectively access the functionality provided. This type of testing examines the overall quality and understandability of the user interface exposed by your system and the workflow that your system embodies; it is vitally important for end-user facing applications, especially when users are not particularly patient or technically savvy.

Again, start the section by explaining Usability Testing: what it is, what it's goals are, how it works in general. Then present a detailed plan for conducting usability testing of your system. Obviously, the testing regime your propose should be based on the specific characteristics of this project (background of end users, novelty of product, nature of consequences of bad design, etc.). Thus start your plan description by outlining these considerations and how they affected your overall thinking on intensity/nature/extent of usability testing you'll do. So again, before you just lay out a testing regime, you need to convince readers that it's thoughtful and well-justified. Discuss the appropriateness of your thoughts on this type of testing with your mentor. You might, for example, leverage focus group techniques to gather qualitative data about the

usage of your system, user studies where you record and analyze user interactions with your software, or expert reviews. In the end, you'll lay out your actual regime: how many expert reviews, realistic user studies, and acceptance testing. Describe exact details for each of these (who, what, how often, etc.), how you'll record the results, and how you'll analyze them to reap insights. Place them all on a timeline that fits within the period you're allotting for Usability Testing.

Conclusion

As always, you'll want a nice conclusion to bring it all together. It should be the usual summary of the entire report, that reviews the main pieces and then makes convincing statements as to why this plan should result in a maximally error-free, functional, and highly usable software product.

Deliverable

1. There is no "draft round" for this document, but it is highly recommended that you discuss a draft with your mentor at a team meeting, to get some feedback before submitting your final version.
2. A final, Software Testing Plan document delivered directly to your team's mentor, in professionally-presented hardcopy. Due on the date listed in the CS486 online course schedule.