Northern Arizona University

Capstone Team Project

Test Plan

HelpTile

Temitope Alaga, John Dance, Joshua Frampton, Jun Rao

CS 486c

Version 1.0

4/13/2016

Table of Contents:

Introduction:
Unit Testing (as needed, about 4-5 pages)
Integration Testing (as needed, about 2-3 pages)
Usability Testing (as needed, about 2-3 pages)
References:

Introduction:

A short summary that briefly describes your overall plans and summarizes activities related to each type of testing you will engage in, as outlined in the sections below. You'll draw the content for this text from the remainder of the document, so this should be the last (and best-written) text you write.

As we conclude much of the development, it is imperative that we employ some testing of our application in order to ensure that it is fully functional. An application that does not work as specified by the requirements is not complete and by employing testing methods we are able to show that our tested portions do or do not perform to specifications. To achieve this, we are employing three main types of testing: Unit testing, integration testing, and user testing. Each of these testing styles are designed to evaluate key failure points of our application. In unit testing we are looking to ensure that individual methods are working as intended. This portion will be the longest because this is where the bulk of the code exists. It is intended to find issues with the low level logic of the application. In our integration we are looking to see that each portion of our application works and communicates as it should. This can be easily visualized as our application is built upon a model view controller architecture. Finally user testing is intended to make sure that the user-facing functionality is working as intended. This will be done using a constructive interaction testing environment where we can get real reviews from end-users. Through all of this testing we can show to the project sponsor that the code that exists is sound and works as defined in the requirements specifications.

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. 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. Also discuss what unit test libraries you are using to streamline your testing activities and any test-related metrics you will be determining (test coverage, for example).

- Test plan
    - Test items
        - The items to be tested are all of the javascript files which located in the js folder.
        - For every javascript file, each of its methods should be tested.
    - Features to be tested
        - The files as described in last section should be tested for compliance based on:
            - Postconditions
            - Preconditions.
        - Both of them are enumerated in each javascript file located in js folder
    - Test deliverables
        - Items to be delivered before the testing begins:
            - The js folder contain source code
        - The following items must be delivered when testing is complete:
            - The Unit Test Report
            - Problem reports
    - Testing tasks
        - Our Unit testing procedure consists of three sets of tasks to be carried out:
            - Before the test
                - The test cases must be written down that includes input values and the output values which we expected.
                - All files must be included because each file and each function is not a single program.
            - During the test
                - The output values are written down, and check the result is match to our expected value or not
            - After the test

        - We need to write a test report which contains the test specification, the expected test values and the output values produced during the test.
    - Environment needs
        - All tests should be run on the same type of system.
        - One of Angular's unit test environments (ie Karma or Jasmine) will be used to structure our tests and keep them well-documented
    - Grading Criteria
        - A test case passes when the produced output of the test program matched our expected output values and no errors have occurred.
        - A test case fails when the produced output of the test program does not match our expected output values and/or when errors occur.

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.

Our goal when creating unit tests is ensure that all of our controller functions and variables are working as intended. Since we are working with AngularJS, all of the functionality is in JavaScript files that represent the controllers. Our views, the display that users of our app will see, will only show the data that controllers contain. Therefore, our views will not undergo unit testing.

AngularJS comes with a basic unit test library that will be used to write the unit tests. There are three main controllers that need to be tested:

- Tile Controller
    - Holds the data for each tile
        - Author
        - Content
        - Comments
        - Collections
    - Contains functions that control the module object displaying the data
        - Adding/Removing to collections

        - Adding comments
        - Sharing via Facebook, Twitter, or Email
- Login Controller
    - Holds user's data
        - Username
        - Password
    - Contains login functionality
        - Logging into user's account
- Home Controller
    - Holds user's requested tiles
        - Tiles the user has made
        - Tiles that match the search the user made
    - Allows functionality
        - Updating tiles when a change in state has been made
        - Allowing users to search for tiles

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

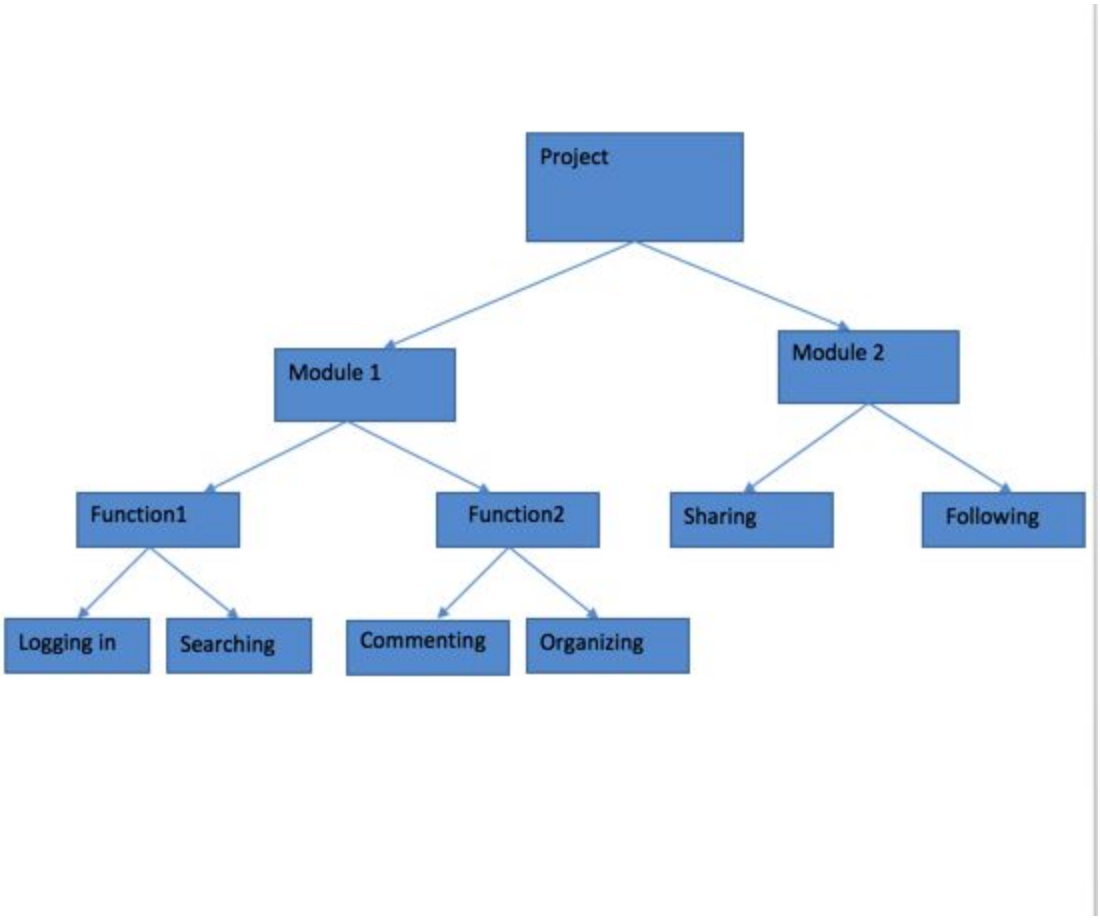
Integration testing is focused on the interfaces between major modules and components, and focuses on whether the interactions and data exchanges between modules take place correctly. At a very simple level, for example, while unit testing may focus on whether a method call returns the correct result, 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 and 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.

Present a detailed plan for testing the integration of major modules in your code: For each integration point, specify what test harnesses you used and how you verified 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.

- Test plan
    - Test items
        - The items to be tested must be present in order to test them.
    - Features to be tested
        - The attributes of the files that cannot be tested by unit testing but passed the unit tests.
    - Test deliverables
        - Items to be delivered before the testing begins:
            - Each unit already passed Unit test
        - Items to be delivered when testing is complete:
            - The Integration Test Report
            - Problem reports
    - Testing tasks
        - Our Integration testing procedure consists of three sets of tasks to be carried out
            - Before the test
                - The input values passed to the unit test and the output values which we expected will be recorded.
            - During the test

        - The output values given by the unit test are recored and compared to the expected output.
    - After the test
        - A test report containing the test specification, the expected test values, and the output values produced during the test will be created.
- Environment needs
    - All tests should be run on the same type of system. The integration test environment is the same as Unit test
- Grading Criteria
    - A test case passes when the produced output of the test program matched our expected output values and no errors have occurred.
    - A test case fails when the produced output of the test program does not match our expected output values and/or when errors occur.

We will use bottom up approach to do our Integration test. As the name suggests, it starts from the lowest unit of the application and gradually moves up.



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.

Present a detailed plan for conducting usability testing of your system, which will vary based on the specific needs and focus of each project. 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 review.

To evaluate the usability of our application we are using user studies. We feel that this method is the best option for our project as we have a reasonably easily reachable target audience and have a very user facing application. The target audience for the application is those who are

familiar with how the HelpTile website functions, and who are in the age range of 18-50. This is backed up by the fact that in order to use the application, one must have previously created an account on the HelpTile website. With this target audience we are able to select from a wide range of test subjects.

In this testing we are planning on using a form of constructive interaction testing. Constructive interactive testing is a method of user testing that involves placing users in pairs and giving them a set of guided instructions via a lab manual. We want to use guided instructions here because we cannot gain as much of the information we are specifically looking to test if we just let the users play with the app instead. With this lab manual we will give context to the test and ask users to walk through tasks in our testing domain.

The key features we are planning to test in our application are as follows:

- Logging in
- Searching for tiles
- Commenting on tiles
- Organizing tiles
- Sharing tiles
- Following tiles

It is important to write tests that use comfortable language to our testers. These users are our end users and so we cannot assume that they know much in the technical field in the context of the HelpTile mobile app. Our instructions must be written in such a way that they are more guidelines that provide context to the instruction. For example, “You begin by logging into the app” is much better than “hit the username field and type in your username, then hit the password field and type in your password, then press the login button”. It is better because it does not use any language that is not commonly used and it does not give away any UI functions, it simply encourages the users to try and figure it out.

References: