The Virtual Office Door | Team Conquistadoors

SOFTWARE TESTING PLAN

James Hauser, Mitchell Hewitt, Nicolas Melillo, David Snow, Tyler Tollefson

Mentor: Dr. Eck Doerry

Clients: Dr. Eck Doerry and Dr. Michael Leverington

Date: 3/21/17

Version: 1.0

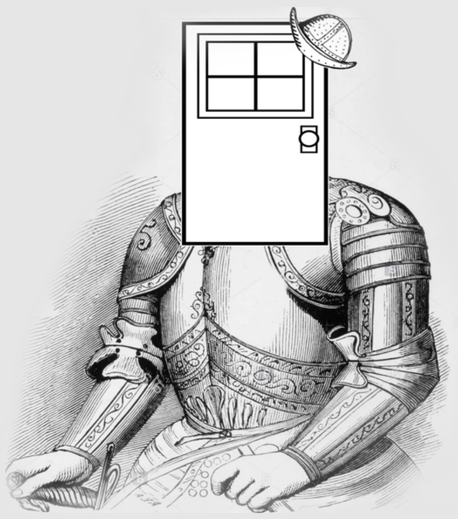

Table of Contents

1. Introduction

Our project, the Virtual Office Door, is meant to serve as the remedy for issues that arise when non-direct communication is a major factor in a workplace over directed communication. From that we need to ask the question, what constitutes non-directed and directed communication? In the scope of our project's problem area, directed communication relates to the use of various technological tools used to relay messages to colleagues, students, etc. Non-directed communication though, takes the form of fliers, sticky notes, bulletins, and other physical media that would be posted on an office door or a cubicle wall. With that being said, our primary goal is to not replace current forms of directed communication, but expand upon current technologies by merging the two forms together. Our Virtual Office Door 's purpose is to effectively be a method of directed/non-directed communication and alleviate the stress that comes from relying on non-directed content.

In the software development world, software testing relates to the idea that, while finishing a piece of software is paramount, testing makes it so that the software actually functions according to previous documentation (requirements document/software design doc). From that we have the different types of tests to be performed: unit, integration, and usability. Each of these types of tests represents a different layer of the application that you are testing, with unit testing being the smallest components possible that you are testing, and then working your way up from there.

Thankfully with our web application, because the UI libraries and backend libraries are straight forward and handle a majority of error checking already, doesn't require as thorough testing as we originally thought. From that we identified that we would need to do the following testing:

- Unit testing for critical widget and backend functions.
- Integration testing for the login page to backend, and the office door page to backend.
- Extensive usability testing for the front page/login and the office door

Our specific software testing plan follows the regime above because of a few factors, and the reasons are detailed below:

1. Our libraries are straight forward and do a majority of the error checking for us. We mainly need to be concerned with how our get/post requests are functioning and some of the other functions we implemented ourselves.
2. Our application is mainly end-user facing. This requires a stricter usability testing section for our testing plan over the other two sections. However, this doesn't mean that unit and integration testing won't be conducted, it is just emphasizing our need to promote ease of use in the application.
3. As we have been developing we have been doing minor error checking to handle certain random cases where we might run into an error, and correcting those so it does not occur further down the line.

The following sections in the document closely represent what we detailed above and reflect our rationale for our testing regime closely. Within the sections we outline the different tests we will be conducting within that testing type and go into further detail about each specific test.

2. Unit Testing

As detailed above, our unit testing isn't the most rigorous testing done in our application, in fact it's quite minimal. Because of that we are only focusing on rigorously testing a small subset of our custom implemented methods as well as get/post requests. With the small amount of unit testing that we are doing we will be creating custom built unit tests and not be using a library, simply to conserve resources and to keep development and refinement on track. Detailed below are the unit tests we will be conducting and how we will be conducting them.

- Testing Get requests
  For testing Get requests, we know exactly the return data types that we will be receiving, so structuring the tests is simple: when the get request is called, does it return the specified data type. This can also be structured with erroneous outputs and inputs but does not need to be with our application.
- Testing Post requests
  For testing Post requests, it requires admin access to the backend and then actually running the tests in the application. How this test will work is the post request will be sent, with data encapsulated, and then the admin will check the backend to make sure the correct data type was received and the data can be processed.
- Testing other minor functions
  There are other functions created within our application, such as functions to return a widget type, delete and item from a list, etc. These will be tested by passing in random widget data or random data in general and then making sure the error checking methods we currently have in place catch data types that don't conform to the functions return type, or their input types.

In the next section we detail the different types of integration testing that we will be conducting as well as the methods we will use to go about the testing.

3. Integration Testing

For integration testing, the primary modules to be tested are the Google API login functionality, the widget functionality, the database functionality, and the communication between these modules. Testing these will be done through the website itself and looking at the results will involve accessing the database through the Django database admin permissions. Certain tests that involve repeating an action many times in a short period of time will be done through an automatic testing software suite such as Selenium or through written scripts. The following are possible tests to incorporate:

<u>Tests to create:</u>

- <u>Uploading the same image multiple times.</u>
  What happens in the database when uploading the same image multiple times?

- <u>Uploading different images sharing the same name.</u>
  What about uploading different images but they all share an identical name?

- <u>Logging in and logging out multiple times with a single account.</u>
  What happens if all you do for an hour is login to your account and logout over and over?

- <u>Logging in and logging out multiple times with multiple accounts at the same time.</u>
  What happens if you do the previous but with multiple accounts?

- <u>Adding a widget and removing multiple times</u>
  What happens when you add a widget and remove it hundreds of times in a short period of time?

- <u>Putting as much information as possible into a widget</u>
  Try putting megabytes of text data within a widget and see what happens when the database receives it.

- <u>Putting database compromising information inside a widget</u>
  If the user knows what kind of database we are using, what happens if you put procedure calls or similar database specific data inside a widget?

- <u>Putting database compromising information inside profile textbox</u>
  If the user knows what kind of database we are using, what happens if you put procedure calls or similar database specific data inside a profile edit textbox.

- Trying to add the same widget multiple times
  The same widget isn't allowed but what happens if you try to add it? Is there a way to forcefully add a widget?

- Making widgets absurdly big or absurdly small
  What happens when you try to make widgets really small or really big? Is the data transmitted properly?

- Putting widgets off screen
  What about when putting widgets off screen? Is that allowed and what happens?

- Logging into the website on separate accounts on separate browsers on the same computer
  What happens when the website recognizes multiple accounts from the same IP?

After creating the above tests we needed to look at how, after everything is integrated successfully, to test our application for ease of use and cleanliness. This is done through rigorous user testing with multiple test end-users.

4. Usability Testing

For this specific type of testing, we decided that the best way to go about it was to have actual users, in this case teachers, set up an office door and test it with their students. From that we could gain feedback from the users on the look of the site, how it navigates, and the look and feel of the widgets. If possible, we want to have a class and professor use the website over a period of time of one to two weeks and at the end of the testing period present a survey with questions. Our primary plan is to create our survey through Google Forms so at the end of the survey access period we can analyze the answers as a whole and look at comments at an individual level.
For the professor the survey will ask questions such as:

- Ease of logging in
- Ease of adding widgets
- Ease of changing widgets
- Usefulness of the website as a whole
- Overall experience using the website

- A question for each main page asking overall appearance of the webpage

Each question will be answer on a 1-5 scale with 1 being very poor and 5 being very good. Along with each question a text box will be provided so the professor can give any desired feedback such as improvements, criticism, and more.

For the students they will also be given a survey but with questions from the point of view of a viewer of doors. They can have questions such as:

- Appearance of the website as a whole
- Appearance of individual web pages
- Usefulness of viewing a door
- Mobile appearance
- Website responsiveness
- Mobile responsiveness

Similar to the survey for the professor, each question will be on a 1-5 scale and along with each question a textbox will be provided so the student can give any further feedback.
Based upon the usability testing response, changes to appearance or functionality can be altered to improve the usability of our capstone product.

From the three types of testing above we will be able to effectively resolve any issues that occur within our application, and make sure that it is deployment ready come the end of April.