SOFTWARE TESTING PLAN FOR THE MODULES OF THE STUDENT SOCIAL NETWORK SYSTEM

For CS 4770 Group B

Prepared by:

Osede Onodenalore, Samuel Ash, Jeff Conway, Maria Aloysius.

March 17, 2017

Introduction 3

Test Items 3

Features to be Tested 4

Approach 4

Item Pass/Fail Criteria 4

Suspension Criteria and Resumption Requirements 5

Test Deliverables 5

Testing Tasks 5

Environmental Needs 5

# Introduction

The purpose of this document is to define our testing strategy for our student social network system. It should serve to portray an overall view of the modules of our system (see srs3 sections 7, 8) at the end of the planning phase. In this document we will utilize white-box and black-box testing, and we will outline our testing criteria, methods for each module/test, and outline several test cases for each module.

# Test Items

In this document, we intend to test all of the modules we have previously outlined in our Software Requirements Specification document (srs3) to find as many bugs as possible to improve the overall quality and integrity of the system. These include:

* Users
* Posts
* Comments
* Groups
* Schedules
* Resumés
* Lost and Found Items
* Polls

All of these modules have a number of variables which must all be included as part of a test for each. For example, a user requires a username, password, email, name, student id, gender, and campus variable. Each of these variables requires a specific type, and some are limited to one of a certain number of predetermined values, for example, user’s campus can only be St. John’s, Corner Brook, Harlow, or Distance. For a complete outline of these variables, see srs3 section modules (page 58).

# Features to be Tested

For each module, we intend to test how it behaves on its own with relation to the view (website view) and the controller (database), as well as test the behaviour of each module in the controller (system code). We also intend to test how several of these modules necessarily interact with one another, for example, every post, comment, group, schedule, resumé, lost and found item, and poll requires a user id as a variable.

# Approach

Our approach to testing the modules of the system will be to test them each one at a time. We will test creating an instance of each, how it stores in the database, and how well it is retrieved from the database. We will also check to ensure that our checks for necessary style of certain inputs (for example, posts cannot be whitespace/empty) are also followed and that appropriate actions are taken when such styles are not followed.

For all modules except user, to create an instance of each, a user id is required as one of the variables included in the module. So, in order to test the creation of each of the other modules, we must necessarily simulate a session as a user of the system.

For testing each module, we will perform our tests on two web browsers, Mozilla’s Firefox and Apple’s Safari.

For each test, a the degree of comprehensiveness required will be relatively low. A user of our system does not require any great knowledge of the system’s modules to interact with it.

For many of our modules, an instance of the module will be required to perform certain actions. For example, a user cannot join a group if no group exists, and a user cannot edit a post if that user has not yet made a post.

# Item Pass/Fail Criteria

For each test, we will outline an expected output which we must achieve in order to say that a test has passed. If this expected output is not achieved, a test will have failed. For example, successful creation of a user will involve viewing of that user object in the database and ensuring all fields are correct, as well as successful redirection in the browser to the login page with a success message displayed.

# Suspension Criteria and Resumption Requirements

Every time testing is beginning or resumed, we must have a clear and up-to-date view of our database, and our system must be currently running via a console command (in our case, node app.js).

For our database, we will be using the application MongoDB Compass, which allows for the simplified view of all objects in a MongoDB database.

When testing is suspended, our system must not be running, i.e., its process must be halted in the console window.

# Test Deliverables

# Testing Tasks

To prepare for each test, the (up to date) code necessary for our system must be found on the computer being used to test each module. Said computer must also have the necessary web browsers, Node.js installed, MongoDB installed, and MongoDB Compass available to conduct each test.

To conduct each test, moderate knowledge of console commands, MongoDB Compass, and use of internet browsers is required.

# Environmental Needs

In order to test our system, after performing the testing tasks above, the computer on which the tests are to be performed must be capable of entering the necessary input, running the necessary programs and commands, and installing the necessary software required. It must also be able to run at a reliable speed to determine whether or not a test takes more time than anticipated.

If any of the above needs, such as software needs, are not yet met, they can be found below.

* Node.js download: <https://nodejs.org/en/>
* MongoDB download: <https://www.mongodb.com>
* MongoDB Compass download: <https://www.mongodb.com/products/compass>