CS3300: Introduction to Software Engineering

Test Plan for Team Chat

Version 1.0: 6/9/15
Team: Web Client

Contents

1 Introduction

1.1 Overview (Executive Summary)

Team Chat is an application providing intra-team communication for projects. It is a client-server application, with a REST API for web, smartphone, tablet, and smartwatch client-side applications.

As the Web Client team, we will be testing the user interface for browser use. The main components we will be testing are user authentication, the ability to manage groups, the ability to send/receive messages, and the ability to share documents. Other areas of testing are compatibility across all popular web browsers and UI responsiveness.

1.2 Assumptions

- The request response JSON is in the expected format with the expected data depending on which request was sent to the server.

1.3 Definitions and Acronyms

- Test Case - Description of the functionality being tested.
- Requirement - Functionality requirement to pass/fail
- Expected Result - Result that we think will happen
- Actual Result - What really happened from the test
- Pass/Fail - Indicator for passing or failing test case
- Notes/TODO - Any additional info that's useful for test result replication or how to correct issue (if any)

1.4 References

- json - www.json.org
- grunt - http://gruntjs.com/api/grunt
- angularJS - https://docs.angularjs.org/guide/unit-testing
- karma - http://karma-runner.github.io/0.12/index.html
- jasmine - http://jasmine.github.io/
- general debugging - www.stackoverflow.com

1.5 Items Not Covered By These Test Cases

As the Web Client team, we will only be testing the web client/server API. We are not responsible for independently testing the smartphone, tablet, and smartwatch modules.

2 Quality Control

2.1 Test Plan Quality

To ensure the thoroughness of our testing, we will make sure we test every piece of functionality on every accessible view. As a rule of thumb, we will not allow anything like buttons that don't actually do anything to exist on a finalized view. There will be test cases for each required piece of functionality. We will keep a tab on which test cases pass and which fail. This way, we know exactly what needs to be corrected to ensure a fully functional product.

2.2 Adequacy criterion

For the overall testing phase, we will cover the functionality stated in the application's requirements. After the functionality is there, we will test the "nice to have" items like browser compatibility and responsiveness. In general, we will call the testing complete when we are able to pass every test case for all of our functional and non-functional requirements.

2.3 Bug Tracking

We will keep an on-going list of bugs in either a text file, a spreadsheet, GitHub issues, or some other project management tool. Each documented bug will show the test case, current status, person who discovered it, and some additional notes. These notes can include steps to reproduce the bug, ideas on how to fix it, and how the bug was finally resolved.

3 Test Strategy

3.1 Testing Process

For this project, we have chosen AngularJS for our JavaScript framework. AngularJS comes with dependency injection built-in. In turn, testing components is much easier, because you can pass in a components decencies as you wish. Components that have their decencies injected allow them to easily be tested at an individual test by test basis. As a result, our code will be highly modularized and demarcated into Model, View and Controllers. We will perform unit testing on every Controller and Angular Service recipe. We will scaffold out our Controllers in a cascading hierarchy; again allowing us to test each piece of code on a test by test basis. In general, we will use unit tests to test both functional and non-functional requirements. However, some requirements, such as usability, are subjective and thus need to be tested by humans and not by software. We will write our unit tests using Jasmine. We will run our Jasmin unit test modules using Karma.

Within our team, everyone has been assigned to be testers. We will each test all of the code that we have written ourselves. Hunter Brennick will oversee all testing as part of his role as Project Manager. Adam Beaton, as the UI Design lead, will ensure that the UI conforms to the project specifications appropriately. Andrew Chow, the Quality Assurance lead, will make sure everyone write code and may verify that code has been properly tested.

3.2 Technology

- Karma for executing our unit tests.
- Jasmine for writing our client-side AngularJS unit tests.
- We will white-box test our UI/UX using all major browsers (Opera, Safari, IE, Firefox and Chrome).

4 Test Cases

Date test performed: TBD
Tester: TBD

Test #	Requirement or Purpose	Action / Input	Expected Result	Actual Result	P/F	Notes
1a	User - Authentication	Correct log in information	Pass			Correct log in information
1b	User - Authentication	Incorrect log in information	Fail			Incorrect log in information
2a	User - Create message	A string of characters	Pass			
2b	User - Create message	Empty string, javascript	Fail			
3	User - Get groups	Returns all the groups currently in.	Pass			
4a	Message - Create document	Appropriate string for document	Pass			
4b	Message - Create document	Empty string, javascript	Fail			
5a	User - Create group	Appropriate parameters for a group object	Pass			
5b	User - Create group	Incorrect parameters for a group object	Fail			
6	Responsiveness	View re-sizes appropriately.	Pass			Run all test cases, must be under a certain threshold
7	Compatible in all browsers	Run through views on all browsers	Pass			Firefox, Chrome, Safari, IE9+, Opera