CS1699 - DELIVERABLE 3: Systems Testing with BDD

Software Under Test: JHangMan, a version of Hangman written in JavaScript

By Liz Davis & Tyler Raborn

Summary

We decided to test JHangman because, as a webapp, it's functionality is entirely user experience driven. Initially, we were going to test a network simulation written in Java, but it's interface was clunky and the application served only to demonstrate various algorithms in graph theory, and as such the user experience was limited. With JHangman, the application logic revolves entirely aroud monitoring the user's actions and reacting to their decisions made in a game setting. As such the number of user scenarios is extremely high, even for the limited functionality provided by the webpage. The tests themselves can be grouped into three functional groups, matching the three main functional modules of the webapp. This includes the high score page, where the application requests high score data from the serverside database, the main game page which fetches a randomly chosen word from the server, and the home page which features a tree-structured menu system for navigation between features of the webapp.

For this assignment, your group will write systems tests (aka acceptance or integration tests) using the BDD model discussed in class. That is, you will write user stories (features) and scenarios in Gherkin (or a similar language, if you prefer), and the step definitions in the language of your project. You should substantially cover a subset of functionality for the project, and note in the "Testing Concerns" section what other aspects you would additionally add for full testing if this were a professional product.

Testing Concerns & Notes

Issues encountered while testing the application primarily revolved around attempting to get Cucumber to work with the application. Cucumber integration with Jasmine proved extremely troublesome, as it required the webapp backend to be written in Ruby utilizing the Sinatra framework. Neither of us had any experience using Ruby or Sinatra, and so programatical integration of the .feature file with our Jasmine unit tests was forgone in the interest of other components of the project.

The JavaScript source code for the webapp is written using Google's AngularJS library. This is located primarily in HangMan/app.js. HangMan/HighScoreService.js also contains a bit of source code. The tests are spread among 3 files to match the 3 modules that they test. HangmanController_test.js tests the functionality of the main game. HomeController_test.js tests the functionality of the main menu. HighScoreController_test.js tests the functionality of the High Score page of the app. The application backend is written entirely in PHP, and is spread across a handful of files, including HangMan.php, Logger.php and Highscore.php.

If this were a professional project, singificantly more tests would need to be performed in able to varify the usability and stability of the application. These would include, but not be limited to, an increased coverage level in the main game logic for such scenarios as SQL injection attacks, the entering of unexpected or unsupported character sets, and the logic utilized by the PHP scripts on the back end.

Format

Every group should have a title page with: * The name of the project under test * The names of the people in the group * The title CS1699 - DELIVERABLE 3: Systems Testing with BDD

First, include a short summary (a few paragraphs) of what you decided to test, and why.

Secondly, add a description of issues you faced when writing these tests and problems you would expect going forward based on your experiences. If any tests you wrote fail, they should be included here. Also note if there are any special steps to get the tests running.

At the end of this section, note where your code (test code and code of project under test) is located.

After this, there should be a printout or screen shot of the test execution results.

There is no need to print out the code. It should be put on a publicly-available site such as Github BY THE BEGINNING OF CLASS.

There shall be AT LEAST 3 user stories per assignment (no matter how many people per group), but there can be more. Each user story should have multiple (at least two) scenarios.

There shall be at least 7 * (number of people on a group) scenarios.

User stories should all follow the Connextra template.

Scenarios should all follow the Given/When/Then template (but note that some scenarios may not require a Given).

Remember that scenarios are FEATURE-level tests; they should discuss things in a way that an intelligent but non-technical user of the software would understand. Remember the differences between scenario tests and specification/unit tests!

Also remember to focus on a particular subset of functionality in order to get good testing coverage.

Grading

- Summary - 5%
- Testing concerns - 10%
- Screen shot or printout of test results - 5%
- User Stories and Scenarios, written in Gherkin - 30%
- Step Definitions - 50%

Reminder that all code (project under test AND test code) should be publicly available, or at least available to me.

Please feel free to email me at bill@billlaboon.com or come to office hours to discuss any problems you have.