CS1699 - DELIVERABLE 3: Systems Testing with BDD

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

GitHub Repository: https://github.com/insomniac34/CS1699_Project3

By Liz Davis & Tyler Raborn

Summary

We decided to test JHangman because, as a webapp, it's functionality is entirely user experience driven. In 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 possible user stories and scenarios is extremely high, even for the limited functionality provided by the webapp. The tests themselves can be grouped into three 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 facilitates the user's guessing of the word, and the "home" page which features a tree-structured menu system for navigation between pages.

Testing Concerns & Notes

Issues encountered while testing the application primarily revolved around attempting to get Cucumber to work with the application. Attempting to get Cucumber working was a process which took hours. Using Intellij, we set up a Maven project with Cucumber integration to manage the build process. Unfortunately, no matter what we tried, the Cucumber JVM would not feed our test program proper command line arguments. We then tried to import the Cucumber JAR file into the Intellij project directly, and this resulted in numerous console errors when attempting to build the project. Eventually it was decided that we would switch to Jasmine and JavaScript. Jasmine was (mistakenly) chosen as a good candidate for BDD because of its verbose, similar-to-English coding style. 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. With minimal time until the due date, it was decided to stick with Jasmine and attempt to mock the BDD process as best as possible.

The JavaScript source code for the webapp is written using Google's AngularJS library. This is located primarily in HangMan/app.js. Hang-

Man/HighScoreService.js also contains a bit of source code. The tests, written in the Jasmine JS testing framework, 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, significantly 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. The backend remains completely untested, and would arguably be the MOST important aspect of a webapp to test, as a user altering code to screw up their game only effects a single instance of the front end, but an attack on the backend could potentially leave the entire webapp crippled and compromised. Another improvement that could be utilized in a professional environment would be the implementation of the PHP backend in Ruby, so as to make access to BDD via the Cucumber environment seamless.

By far the biggest mistake/regret was not using a true BDD framework. It was discovered shortly before the due date that Behat(http://docs.behat.org/en/v2.5/), a BDD framework for PHP that mocks the usage style of Gherkin, could have been used to test the application backend via true BDD.

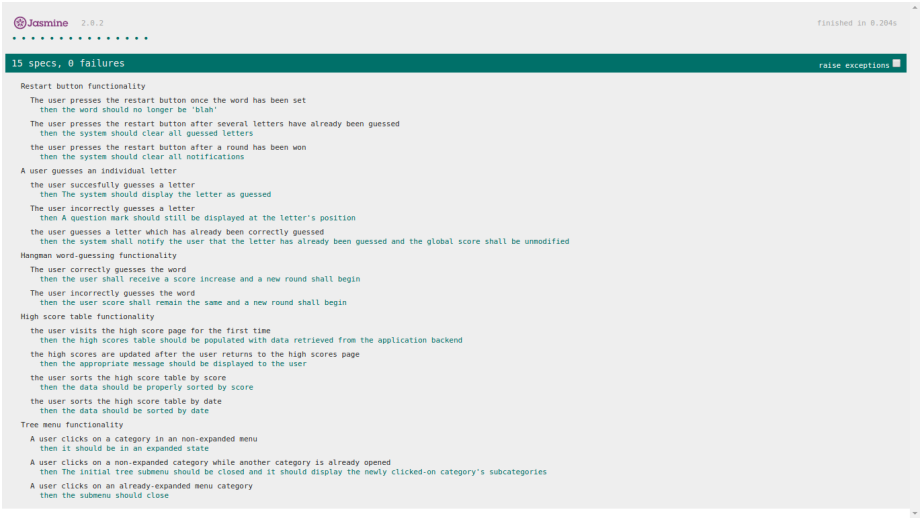

Figure 1: Screenshot of Unit Tests in Jasmine

Installation of the Project

In order to set the project up locally, the following software is required: * Apache with PHP enabled * MYSQL * Following the above software requirements, you must then execute Hangman/init.php to create the database. * THEN, you must execute the following queries from within MYSQL: * use test1; * CREATE TABLE Scores (id int NOT NULL AUTO_INCREMENT, scoreDate datetime NOT NULL, score int, PRIMARY KEY (id));