Project Progress Report

Sanil Shah | UNI: ss4924

Topic: Analysis and evaluation of JavaScript testing frameworks

Overview

I plan to work on my project independently and build on the topic I explored while researching and writing my midterm paper. The project will involve understanding several JavaScript testing frameworks in greater detail. The project will first implement a JavaScript application that will interact with DOM elements so as to mimic a real world application. Unit tests for this application will then be implemented using three different test frameworks. The project will aim to analyze the developer effort and test coverage obtained from implementing unit tests in each of the frameworks to better understand the most effective framework or tool to use.

Project Details

I will start by writing a simple tic-tac-toe application where a two human players can play against each other on the same computer. Once the application is written along with the UI and DOM interactions to allow users to properly play the game, I will add unit tests for the application using three different test frameworks. The project will concentrate on Mocha, Jasmine and Jest since they seem to be the most commonly adopted test frameworks today. While implementing these test frameworks, I will measure the time taken to write the unit tests, setup the test framework and how long the test suite takes to build and run from scratch. Based on the features available in each framework I will also add integration, functional or snapshot tests to ensure that the UI looks correct and the application behaves correctly when actually being used. This will be done without integrating with Selenium if possible in the interest of time and brevity.

The project report will compare the differences in productivity and developer effort from implementing tests in each of these test frameworks. It will also measure code coverage to understand how effective testing techniques were across test frameworks. Most new JavaScript developers shy away from testing because of the setup time needed to get these frameworks up and running and properly integrated with their code, so this project will measure and evaluate the this metric as well. As a first time user for each of these test frameworks, I believe that my implementing them from scratch for a new application will provide an unbiased benchmark.

Deliverables

The following code will be submitted using GitHub. They will be submitted under a new project here: https://github.com/sanils/tic-tac-toe

- Tic-tac-toe application written in JavaScript
- Tic-tac-toe front end and UI in HTML and CSS that interacts with the JavaScript
- Unit tests implemented using Mocha
- Unit tests implemented using Jasmine
- Unit tests implemented using Jest
- Integration or end-to-end tests where applicable
- Snapshot tests implemented in Jest

Further, as part of the deliverables, the project will present a detailed comparison of the metrics discussed above in the project details section.

Progress

Since this project is evaluating JavaScript testing frameworks that have been developed by others, I will answer the questions listed in that part of the assignment prompt.

- *Systems being evaluated:* Jasmine, Mocha and Jest testing frameworks
- *Why chose them:* I chose these three testing frameworks based on the research done for the midterm paper. The developer survey conducted for the midterm paper indicated that these three test frameworks were the most popularly used ones for JavaScript testing. Further, I chose frameworks that I had never used before to provide unbiased results for set up times and to understand the difficulty of learning to use these frameworks
- *Novel or unique:* There are plenty of articles and blog posts comparing the functionality offered by various JavaScript testing frameworks. They also offer simplistic examples on how to write basic tests. However, there doesn't seem to be an in depth analysis of the actual time it takes to setup and configure these tests for a small to mid-sized project that is more than a playful example. This project will provide a benchmark to help compare setup and implementation times for the three test frameworks mentioned above. This will provide a deeper understanding of which framework is easiest to use and hence help improve developer productivity and test coverage by decreasing the reliance on manual testing.

- *User community:* This study is primarily targeted at JavaScript developers who heavily rely on manual testing today and have no form of unit or integration tests. This could be beginners or intermediate developers as indicated by the survey in my midterm paper. The value offered by this project is two-fold. First it will provide more robust examples by actually creating a mid-sized project which will then be sufficiently tested using all three frameworks. These can be used as a starting point for developers looking to test their code. Secondly, the analysis of the time taken to setup and configure the frameworks, the number of web searches required to get them working, the ease of implementation, and time taken to run the test suites can be used by these developers to determine which test framework is easiest to use and most suitable for their projects.
- *Benchmarks and metrics:* The primary metrics I plan to collect are as follows:
    - Setup time - This is essentially the time it takes to get the most basic unit test running.
    - Implementation time - This is the time it takes to write a complete suite of tests for the tic-tac-toe program described above.
    - Running time - This measures the time it takes to execute the full suite of tests.
    - Coverage - This will measure code coverage where possible using the framework itself if it is built in, or using a third party tool to measure coverage using the different frameworks. Ideally this should be the same across all frameworks, since I should be able to write similar unit tests for each of them.
    - Search queries - I will try and keep track of the number of Google searches I use in implementing unit tests in each of these frameworks. This will provide yet another data point to indicate the level of difficulty in using these frameworks.

As far as progress I've made so far, I'm still working on the tic-tac-toe program itself. I aim to have this complete by April 15th at which point I will begin integrating with the test frameworks listed above. The process of implementing the tests will itself act as the analysis since I will measure the various metrics during the process of using the test frameworks.

Demo

For my demo I plan to be able to show the following in class:

- A demo of the actual tic-tac-toe game. I will demonstrate a few different scenarios that may occur such as either player winning, a tie game, attempting to play on an already occupied square and any other "error" scenarios students in class may be able to think of. This will demonstrate both that the code well tested and the unit tests were effective in catching any bugs that may have existed before the demo.
- A demo of executing the Jasmine test suite. The time it takes to run these tests should be similar to the "running time" metric in my analysis of the various test frameworks .

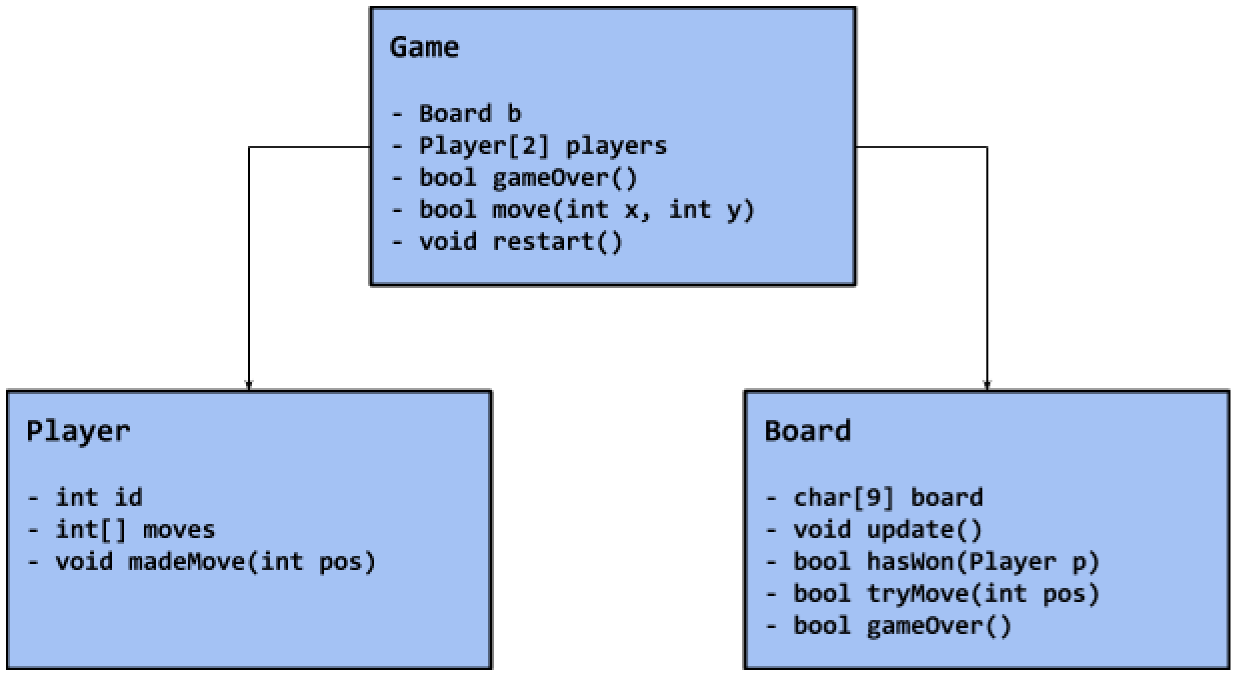
- A demo of executing the Mocha test suite.
- A demo of executing the Jest test suite.
- If time permits, I will modify one of the tests in each of the test suites to demonstrate that the tests fail if the expectations or assertions are changed.
- Lastly, I will present my analysis of the test frameworks and make a recommendation for the one that I found easiest to set up and use to implement robust tests for the tic-tac-toe program.

Tools

The following third party and open source frameworks will be used for this project.

- Jest: https://facebook.github.io/jest/
- MochaJS: https://mochajs.org/
- Jasmine: https://jasmine.github.io/
- Istanbul (Coverage): https://istanbul.js.org/
- Material Design (CSS): https://getmdl.io/index.html

Diagram


Game
- Board b
- Player[2] players
- bool gameOver()
- bool move(int x, int y)
- void restart()
Player
- int id
- int[] moves
- void madeMove(int pos)
Board
- char[9] board
- void update()
- bool hasWon(Player p)
- bool tryMove(int pos)
- bool gameOver()