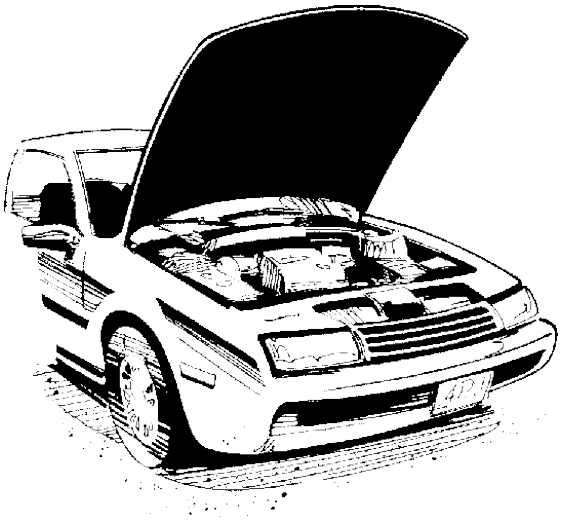
INFSCI 2545 – Deliverable 3: Web Testing with Behavior Driven Development

Joseph David Bender

jdb117@pitt.edu

Testing the Hoodpopper Ruby Web Application



Issues Faced While Coding

This assignment was much more comprehensive than the in-class exercise with Reddit. For this project we had to formulate the user stories and scenarios ourselves. Therefore, we were in complete control of the factors influencing the behavior driven development. We couldn't just dive right into coding. First, we had to tinker around with the Hoodpopper web application to get a feel for its functionality and how it worked. I didn't start programming until I had a grasp on how all three buttons would manipulate the text input. There was a test where I stored the result of a parse in a string, clicked "Back" so I could enter a different input, and then tried to compare the resulting parse to the first execution. However, I think clicking "Back" messed up my variables or driver somehow. Instead, I created a second driver and just performed two separate loads of the webpage. I then compared the resulting string variables for my desired assertion. In addition, I receive some CSS errors in the console upon execution. All the tests still pass, and I remember us running into the same inconsequential feedback during the in-class exercise. However, the errors are still presented. I also found it a bit difficult to come up with three scenarios for each user story, but was eventually able to. It was just challenging to grasp the Ruby architecture enough to come up with meaningful tests. In the end though, I enjoyed this assignment and find selenium web testing useful, relevant, and informational.

Problems I would expect going forward

I would be hard pressed to think of more tests to perform for an application with such a simple functionality. There are only so many observations to be made on the operations performed by Hoodpopper. In addition, I think there are not many typed of users who would utilize this program. It is very useful for the meta-analysis of Ruby code, but beyond that the usefulness is limited. Therefore, generating more user stories would be troublesome. If this were going to be a professional product, I would maybe include additional testing to ensure reliability. There may be some characters that the user could enter to break the Hoodpopper. Some languages have special code that would exit the typical syntax and result in an error. There could be additional methods that insert uncommon characters into the text area to see if the app can handle them.

Failed Tests:

None of the tests fail. All 9 tests are completed successfully.

Special Steps to Initiate Tests

I don't believe there are any special circumstances to run my code. The webpage is loaded automatically into the driver in the setUp() method. I import the JUnit 4 library, selenium standalone server, selenium java, and Mockito jar files to the build path.

USER STORIES

1. User Story #1 – Data Analyst
    a. As a data analyst, I want to investigate tokenization of data, so that I can better understand how Ruby code is segmented and broken down
2. User Story #2 – Computer Language Specialist

a. As a computer language specialist, I want to investigate the grammar rules of Ruby, So that I can fully grasp the complexities of the abstract syntax tree

3. User Story #3 – Bytecode Developer
    a. As a bytecode developer, I want to understand how to translate the abstract syntax tree, so that I can better interpret the resulting executable code for the Ruby YARV virtual machine

SCENARIOS

1. Scenarios for User Story #1 – Data Analyst

a. Scenario 1.1
    i. Given that the website is loaded
    ii. When an integer variable is declared, and the code is tokenized
    iii. Then an ":on_ident" token should exist

b. Scenario 1.2
    i. Given the webpage is loaded
    ii. When a variable is declared, and the same variable is then printed with the "puts" command
    iii. Then both the declaration and put statements should be listed in the tokenization as the same, with ":on_ident"

c. Scenario 1.3
    i. Given the webpage is loaded
    ii. When two identical declarations, but one with whitespace, are entered
    iii. Then the one with whitespace will have more tokens of "op_sp"

2. Scenarios for User Story #2 – Computer Language Specialist

a. Scenario 2.1
    i. Given that the webpage is loaded
    ii. When two declarations are parsed that have the same operation, but different amounts of whitespace
    iii. Then the length of the parse should still be identical, regardless of whitespace

b. Scenario 2.2
    i. Given the webpage is loaded
    ii. When the only text parsed are a combination of spaces and newlines
    iii. Then the parser shall not construct an AST, but return "void_stmt"

c. Scenario 2.3
    i. Given the webpage is loaded
    ii. When two variables are declared
    iii. Then you can see the variables being assigned as integers using "@int" in the abstract syntax tree

3. Scenarios for User Story #3 – Bytecode Developer

a. Scenario 3.1

i. Given the webpage has loaded
ii. When two declarations and an addition are performed
iii. Then two "putobject" lines, and a "opt_plus" command, will be constructed in the bytecode for the Ruby YARV virtual machine

b. Scenario 3.2
i. Given the webpage is loaded
ii. When four variables are declared
iii. Then the resulting compiler table size should equal 5

c. Scenario 3.3
i. Given the webpage is loaded
ii. When four variables are declared using the mathematical operations of addition, subtraction, multiplication, and division
iii. Then all four declarations will be identifiable in the compiler code as "opt_plus", "opt_minus", "opt_mult", and "opt_div"

Screenshot of Successful Tests