October 3, 2017

Deliverable #2 The Testing Process

1. TESTING PROCESS

A. The testing process will begin with manual testing, during which individual methods will be tested for expected outputs via the terminal. After our initial approach here with five test cases, we will then be able to expand our testing selection to encompass a larger part of Sigmah to ensure that all of their source code performs as it is intended.

2. REQUIREMENTS TRACEABILITY

A. truncate (Number n): This method truncates the parameter input, n, to two decimal places.

B. truncate (Number n, int decimals): This method truncates the parameter input, n, to parameter input, decimals, length.

C. truncateDouble (final Double value): This method truncates the parameter input (Double), value, to two decimal places. This method calls truncate.

D. ratioAsString (Number n, Number in): This method returns parameter input, n and in, as a ratio in the format of a percent. This method calls ratio and truncate.

E. ratio (Number n, Number in): This method returns a ratio as a double from the parameters n and in.

3. TESTED ITEMS

A. The tested items consist of several methods within the NumberUtils.java file. Some of these methods to be tested are two instances of truncate, truncateDouble, ratioAsString, and ratio. Each of these methods performs a crucial task to ensure that user inputs match the expected input, and if not they are truncated to match.

4. TESTING SCHEDULE

A. October 3 --> Specified 5 test cases (*.txt).

B. October 10 --> Performed our 5 test cases with Sigmah's code (standalone).

C. October 17 --> Ran into dependency issues with our Drivers.

D. October 23 --> Solved dependency issues with our Drivers, finalized scripting code to run all 5 test cases.

E. October 31 --> Have 5 test cases plus testing framework operational.

F. November 7 --> Have 15 test cases developed.

G. November 14 --> Have 25 test cases developed and ready for presentation.

H. November 21 --> Have 5 fault injections ready that will cause at least 5 tests to fail, but not all; ready for presentation.

I. November 28 --> Have final write up, PowerPoint, and poster ready as well as the 25 test cases ready to present.

5. TEST RECORDING PROCEDURES

A. The test recording procedure states that tests must be systematically recorded. As of this deliverable we have specified 5 of our test cases and we currently plan to have the outputs (pass or fail) be recorded to a text file of some sort with details of the test case and whether it passed or failed. Each of our test cases will be in a text file with the test case number, the purpose of the method tested, the name of the class tested, the name of the method tested, sample input, and expected output. Our script will read in from this text file and perform the test, comparing the actual to expected output and reporting whether the test case passed or failed.

6. HARDWARE AND SOFTWARE REQUIREMENTS

A. Java JRE, Java JDK 1.6, PostgreSQL 9, Maven 3, and Git 2.7 are required to compile and run the Sigmah software.

7. CONSTRAINTS <-- THIS WILL BE ONE WE CHANGE A BIT (NOAH AND ANDREW)

A. Our constraints in this testing process will be mostly due to time constraints with our group members. All three of us are seniors here at the college and we have very different schedules. Between homework and projects for all of our other classes, and one of our group members having two kids and a part-time job, we have already experienced issues with finding a time to schedule meetings. However, we have already made plans to accommodate these constraints by having a very in-depth and regularly updated plan to split up the workload and meet up remotely if necessary. Some other restraints that are worth mentioning are the limitations of the size of our group, as such we are limiting our testing framework to 25 test cases for now.

8. SYSTEM TESTS

A. Test Case 1

i. Test #0001
ii. This method truncates the parameter input, n, to two decimal places.
iii. NumberUtils.java
iv. truncate(Number n)
v. 3.883
vi. 3.88

B. Test Case 2

i. Test #0002
ii. This method truncates the parameter input, n, to two decimal places.
iii. NumberUtils.java
iv. truncate(Number n)
v. 3.888
vi. 3.88

C. Test Case 3

i. Test #0003
ii. This method truncates the parameter input, n, to two decimal places.
iii. NumberUtils.java
iv. truncate(Number n)
v. 3
vi. 3.0

D. Test Case 4

i. Test #0004
ii. This method truncates the parameter input, n, to two decimal places.
iii. NumberUtils.java
iv. truncate(Number n)
v. string
vi. java.lang.NumberFormatException: For input string: "string"

E. Test Case 5

i. Test #0005
ii. This method truncates the parameter input, n, to two decimal places.
iii. NumberUtils.java
iv. truncate(Number n)
v. 3.88
vi. 3.88