Top of Form

Bottom of Form

**CS 1632 - Software Quality Assurance**

Spring Semester 2019

ASSIGNED 18/19 MAR (MW/TH); 1/2 APR (MW/TH)

**Deliverable 3**

For this assignment, you (NOT with a partner) will create a simple web application and test it using automated systems-level testing with Katalon. This web application will accept three parameters, entered via textboxes, on the main page. These three parameters will be the symbol you wish to use for "true", the symbol you wish to use for "false", and the size of the truth table (in terms of 2 ^ x, e.g., entering three will create three binary digits which means the size of the truth table is 2 ^ 3 == 8). You will then display the truth table for the operations AND, OR, NAND, and NOR.

If invalid data is entered (i.e., a multi-character true or false symbol, the same character used for true and false, or anything other than an integer >= 2 for the size), an error page should be displayed.

If the user attempts to go to a URL other than one specified, a separate error page specifying the issue, along with a 404 status code, should be displayed.

I am not interested in how "pretty" you can make the output; there is no need to add CSS or style the page in other ways. For those of you who have done web development before and for whom this may seem painful, simply imagine that you have time-travelled back to 1995, before CSS was invented. This is also good practice in case you do end up time-travelling to the mid-1990s (the first version of Ruby was released in 1995, so you will still be good on that score, although the documentation was originally written only in Japanese).

Requirements for this program are in the requirements.md file in this directory. In case of ambiguity, please see the sample output as an example of what to display and how the system should work.

All code and tests should be on GitHub in a private repository accessible to me and the TA.

Your repository MUST be named D3.

**Format**

You should turn in a title page with:

* The title "CS 1632 - DELIVERABLE 3: Automated System Testing of a Web Application"
* Your name and your GitHub username
* The URL of your code and tests on GitHub

ENSURE THAT THE THE TA AND I ARE ADDED AS COLLABORATORS ON GITHUB! My username is laboon. The TA's username is:

* M/W 11:00 - 12:15 : Tianyi Cui, GitHub username TyaCUI
* M/W 1:00 - 2:15 : Akhil Yendluri, GitHub username aky13
* T/H 9:30 - 10:45 : Xiaoyu Liang, GitHub username Grade-CS1632

**You will lose an automatic 10 points if either I or the TA cannot access your repository.**

Before turning this assignment in, I HIGHLY recommend you double-check that all of the code that you think is committed is in fact committed by (1) looking at the code on the GitHub web interface and (2) cloning it down to a new directory on your machine and trying to run it from there.

Add a short ( < 1 page ) description of issues you faced when writing this code and tests. If any tests you wrote fail, they should be included here, along with why you think that they are failing.

After this, ON A SEPARATE PAGE, include a screenshot of the executed unit tests (this means a SCREENSHOT of the Katalon tool after execution of the test suite). If a test doesn't pass, it should be included in the concerns section above. Ideally, for a perfect grade, all tests should be green (passing). However, if you have what you think is a valid test and it is not passing, I would prefer that you include a defect report (in the proper template!) rather than just deleting it. Knowing that a defect exists and reporting it is much better than having it discovered by the customer (me)!

There is no need to print out the code itself. It should be on GitHub BY THE BEGINNING OF CLASS.

There should be a minimum of sixteen (16) Katalon tests, checking all requirements (as measured by the traceability matrix). Note that this is an ABSOLUTE MINIMUM value. You may have more tests if you like.

Keep in mind some of the things we learned when doing manual and unit testing; you should be cognizant of equivalence classes, boundary values, etc. and focus on them.

For this project, you should endeavour to get a good sampling of different equivalence classes and success/failure cases.

You should be able to handle any invalid input coming into the program and should test for such situations. Invalid input could include bad parameter values, bad parameter names, bad URLs, etc. It is up to you to think of proper edge and corner cases and how to respond to these failures!

**Extra Credit**

There are two possibilities for extra credit, for a total of +7 points on this assignment. Please indicate on the cover sheet if you have done extra credit 1, 2, or both.

1. (+2 points) Add the following two logical operators - XOR and SINGLE to the truth table, along with an additional *four*Katalon tests for testing these. Recall that for any set of values n, XOR is defined as being true if an *odd* number of values are true. For example:

* XOR(TFT) is false (two trues = even)
* XOR(FFT) is true (one true = odd)
* XOR(FT) is true (one true = odd)
* XOR(TT) is false (two trues = even)
* XOR(TFTFTFTFT) is true (five trues = odd)
* XOR(FFTFTFTFT) is false (four trues = even)

The SINGLE operator returns true if and only if a single value is true and all others are false. For example:

* SINGLE(FFFF) is false (zero trues)
* SINGLE(FFTF) is true (one true)
* SINGLE(FFTT) is false (two trues)
* SINGLE(TTTT) is false (four trues)
* SINGLE(FT) is true (one true)

1. (+5 points) Add the ability to save your truth tables as a particular name and access them later. For example, you may save a truth table as foo, then access it at /saved/foo and it will display the saved table just as if you had entered the initial parameters. This does not have to be persisted to disk (although it may be!); it is allowable to store it in memory (and thus lose it after the Sinatra server is shut down). However, you should show an appropriate error message if the truth table with a particular name does not exist.

You must also include an additional five Katalon tests for this.

If you do either of the extra credits, you MUST mark down on your cover page which extra credit problems you did (1, 2, or both 1 & 2).

**Grading**

Just like in D1, you should line up requirements with test cases and develop a traceability matrix to let me know which Katalon tests line up with which requirement.

\* Summary and Testing concerns - 10%

\* Screen shot of test results - 5%

\* Functionality - 40%

\* Katalon tests - 40%

\* Traceability matrix - 5%

**Tips**

1. Although reloader will work on the main file you call (i.e., truth.rb) and any views, it will NOT reload any helper files you may have require\_relative'd! You will have to restart the Sinatra server to reload them.
2. You will find this assignment much easier if you learn the <table> HTML tag along with its related tags, <tr>, <td>, <th>.
3. Remember that erb allows you to run any Ruby code you want in the .erb file. However, I recommend you do most of the "computational heavy lifting" elsewhere, because 1) adding the <% and %> gets annoying and 2) it's good practice to separate your calculations and data ("model") from how it is displayed ("view").