Deliverable 2: Chapter 2

Edit New Page

Jump to bottom

Jack edited this page now · 1 revision

Test Plan

Testing process

1. Find Testable Method
2. Import method to the TestAutomation/project/src/ directory
3. Modify method as needed to make it able to interact directly with the test environment.
4. Determine valid / invalid inputs for method
5. Write tests for method
6. Write test cases for tests in test environment
7. execute `./scripts/runAllTests.sh`

Requirements traceability

If any tests fail, requirements are not being met

Tested items

Any js file which has standalone functions are available to be tested. We will continue to add tests and test files until the project ends.

Testing schedule

Whenever tests are due, we will have done them by.

Test recording procedures

Utilization of recording frameworks within nodejs allows an html record of the test process to be generated and stored automatically on test run.

Hardware and software requirements

Hardware should be fairly permissive, but software is linux-based. Most shell scripts will work even in the WSL, but unfortunately the automatic file opener will run into issues if a cross-platform web-browser is not set up within WSL settings.

Constraints

Time and energy; we will write tests until we no longer have to; such does not result in a thoroughly tested software

System tests

As this project is unit testing, system tests are outside our purview. Sadly, as they seem pretty cool; but happily, as the system is rather complicated and as such would be pretty hard to write tests for.

Test Layout:

Test Case Files: Standard Layout

1. Test ID:

   ID is descriptive to the name of the file being tested e.g. 'units_test_16' is the 16th test of the 'units' file

2. Requirements being tested:

   short description of why this test exists

3. Component being tested:

   name of file (or files) containing source code being tested

4. Method being tested:

   name of function (or functions) which are called within the test

5. Function:

   summary of the features of the function(s) called

6. Input:

6. Input:

   input value

7. Expected output:

   expected return value note: this value is what the method *should* return; if it fails, blame the developers

Test Case Executables Interface: BDD

```
describe('Array', function() {
  before(function() {
    // ...
  });

  describe('#indexOf()', function() {
    context('when not present', function() {
      it('should not throw an error', function() {
        (function() {
          [1, 2, 3].indexOf(4);
        }.should.not.throw());
      });
      it('should return -1', function() {
        [1, 2, 3].indexOf(4).should.equal(-1);
      });
    });
    context('when present', function() {
      it('should return the index where the element first appears in the array',
function() {
        [1, 2, 3].indexOf(3).should.equal(2);
      });
    });
  });
});
```

https://mochajs.org/#interfaces

Testing Framework Environment Details

Operating System and Testing Framework

- Linux
- NPM (node js package manager; essential for building and testing js code)

- Mocha (one of many available test case frameworks within nodejs; selected for being simple to use)
- Mochawesome (a lovely html file output extension to mocha)