|  |  |  |  |
| --- | --- | --- | --- |
| Mocha | Has been used to automate and document unit testing for the site/application | 0.16.1 | Test |
| Selenium | Is used to automate application testing via the browser | 2.90 | UI Test |
| WAT | Used to conduct 508 testing on the application | 2012-09-13 | 508 Test |

The following code strings were developed as unit test scripts.

**Mocha**

var should = require('should');

var assert = require('assert');

var request = require('supertest');

var httpReq = require('request');

describe('Routing', function() {

var url = 'http://foodsafety.ibacorp.us';

//var url = 'http://www.google.com';

// within before() you can run all the operations that are needed to setup your tests.

// when you are done, call done().

before(function(done) {

// do the pre test setup

var date =new Date().toISOString().

replace(/T/, ' '). // replace T with a space

replace(/\..+/, '')

console.log('\* Test Started at ' + date)

done();

});

// use describe to give a title to your test suite, in this case the tile is "FoodREcallTests"

// and then specify a function in which we are going to declare all the tests

// we want to run. Each test starts with the function it() and as a first argument

// we have to provide a meaningful title for it, whereas as the second argument we

// specify a function that takes a single parameter, "done", that we will use

// to specify when our test is completed, and that's what makes easy

// to perform async test!

describe('FoodRecallTests', function() {

it(**'should return error if FoodSafety Site is NOT Alive**!', function(done) {

// once we have specified the info we want to send to the server via POST verb,

// we need to actually perform the action on the resource, in this case we want to

// GET and we want to send some info

// We do this using the request object, requiring supertest!

request(url)

.get('/')

// end handles the response

.end(function(err, res) {

if (err) {

throw err;

}

// this is should.js syntax, very clear

res.should.have.status(200);

done();

});

});

it(**'should return error if can not fecth the deafult page'**, function(done){

var body = { };

request(url)

.get('/#/default.htm')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

res.should.have.status(200);

});

});

it(**'should return error if can not fecth the contact page'**, function(done){

var body = { };

request(url)

.get('/#/contact.htm')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

//console.log(res);

res.should.have.status(200);

});

});

it(**'should return error if can not fecth the about page'**, function(done){

var body = { };

request(url)

.get('/#/about.htm')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

res.should.have.status(200);

});

});

it(**'should return error if can not select region to check food recalls'**, function(done){

var body = { };

request(url)

.get('/#/form/Viginia/va')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

//console.log(res);

res.should.have.status(200);

});

});

it(**'should return error trying to post and get data from openFDA**', function(done) {

var profile = {

params: {

product\_description: 'Cheese',

recalling\_firm: 'Kraft' },

distribution\_pattern: 'VA',

status: 'Ongoing'

};

// once we have specified the info we want to send to the server via POST verb,

// we need to actually perform the action on the resource, in this case we want to

// POST on /api/profiles and we want to send some info

// We do this using the request object, requiring supertest!

request(url)

.post('/foodQuery')

.send(profile)

// end handles the response

.end(function(err, res) {

if (err) {

throw err;

}

// this is should.js syntax, very clear

//console.log(res.body);

res.should.have.status(200);

done();

});

});

it(**'should return error if can not fecth the recalls page**', function(done){

var body = { };

request(url)

.get('/#/recalls/')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

//console.log(res);

res.should.have.status(200);

});

});

it(**'should return error if can not fecth the about page'**, function(done){

var body = { };

request(url)

.get('#/recall/1')

.end(function(err,res) {

if (err) {

throw err;

}

done();

// this is should.js syntax, very clear

res.should.have.status(200);

});

});

});

});

**WAT and Selenium** open items from unit test closed in version 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| An interactive element or function cannot be accessed or activated by keyboard | 21(a) | B01 | Not Compliant | Cannot tab thru "food", "brand" and "all".  See picture below. |
| A Web image does not have an ALT, TITLE or ARIA attribute | 22(a) | B07 | Not Compliant | No alt for "Food Recall Safety" and "Powered by IBA" pictures.  Get message 2 image without alt attributes. See picture below |
| The contrast ratio is less than 4.5:1 for content background and foreground colors | 31(b) | B09 | Not Compliant | 1. "Food Recall Safety" has a color contrast of 4.49:1. See picture below.  2. "Home",  "About" and "Contact" links have a color contrast of 4.49:1. See picture below.  3. "food",  "brand" and "North Dakota" and "all" links have a color contrast of 4.01:1. See picture below. |
| There is no meaningful page title in plain language | 31(a), 31(b) | B12 | Not Compliant | Should say "Food Safety Recall - Results". It currently says "Food Safety Recall". See picture below. |
| There is no method to skip repetitive content | 22(o) | B05 | Not Compliant | No Local anchors found.  See picture below. |