# Angular JS Unit test framework –
## Karma jasmine configuration steps

## VERSION CONTROL

| | |
|---|---|
| Prepared by: | S.Kalaiyarasu |
| Date: | 19/11/2015 |
| Reviewed and Accepted by: | Pattar. Monesh |
| Date: | 26/11/2015 |
| Approved by: | Baskaran.Varadarajan, D.A.Soundararajan |
| Date: | 29/12/2015 |

## VERSION HISTORY

| Date | Version | Author | Description |
|---|---|---|---|
| 19/11/2015 | 0.1 | S.Kalaiyarasu | Initial draft version |
| 26/11/2015 | 1.0 | Pattar. Monesh | Final Reviewed |
| 28/12/2015 | 1.1 | Baskaran.Varadarajan | Final Approved |
| | | | |
| | | | |
| | | | |

# Contents

# 1. INTRODUCTION TO UNIT TEST CASE FRAMEWORKS

Unit testing involves **breaking your program into pieces**, and subjecting each piece to a series of tests.

Unit testing simply verifies that individual units of code (mostly functions) work as expected. Usually you write the test cases yourself

Some of unit test case frameworks:

1. Junit
2. Mocha
3. UnitTesting
4. JSpec
5. Jasmine.

Jasmine is one of the framework which will be used to write test cases for javascript based files. Once done writing the test case using jasmine framework, that you can run using the karma test runner.

## 2. INTRODUCTION TO KARMA JASMINE FRAMEWORK USING YEOMAN

**Karma:**

Karma is a test runner for JavaScript that runs on Node.js. It is very well suited to testing AngularJS or any other JavaScript projects. Using Karma to run tests using one of many popular JavaScript testing suites (Jasmine, Mocha, QUnit, etc.) and have those tests executed not only in the browsers of your choice, but also on the platform of your choice (desktop, phone, tablet.) Karma is highly configurable, integrates with popular continuous integration packages (Jenkins, Travis, and Semaphore) and has excellent plugin support.

**Jasmine:**

Jasmine is an open source testing framework for JavaScript. It aims to run on any JavaScript-enabled platform, to not intrude on the application nor the IDE, and to have easy-to-read syntax.

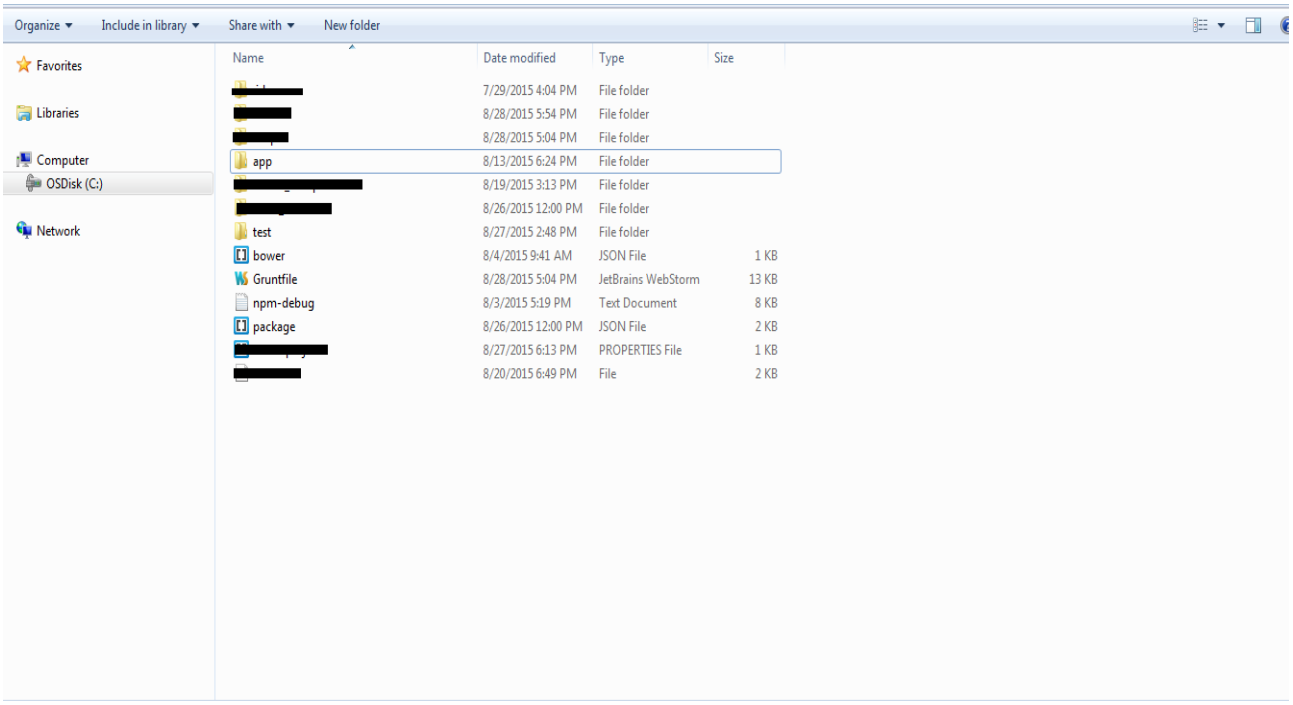
**Yeoman:**

Yeoman helps you to kick start new projects, prescribing best practices and tools to help you stay productive.

To do so, we provide a generator ecosystem. A generator is basically a plugin that can be run with the `yo` command to scaffold complete projects or useful parts.

# 3. FOLDER STRUCTURE

Yeoman installation will provide us the default MVC folder structure for our project. Which includes karma and jasmine by default in our configuration. Once conifgured the yeoman, the folder structure will be looks like as shown below.




As presents in screen shot above, **app, test, bower.json, gruntfile.js and package.json** are the important files that yeoman will create while configure it.

Test folder will contains all the unit test case file for respective source file present under app folder.

**Note:**

Running the **'YO'** command in your command prompt under specific path, will generate the folder structure as shown above.

# 4. KARMA JASMINE CONFIGURATION FILES

## Karma.conf.js:

- The one important file required for testing jasmine and karma is karma.conf.js

This file contains all the dependencies that are belong to our angular js application module. While running grunt test command, this will loads all the dependencies files presents in karma.conf.js file and cross check with index.html.


**Sample Karma.conf.js file**

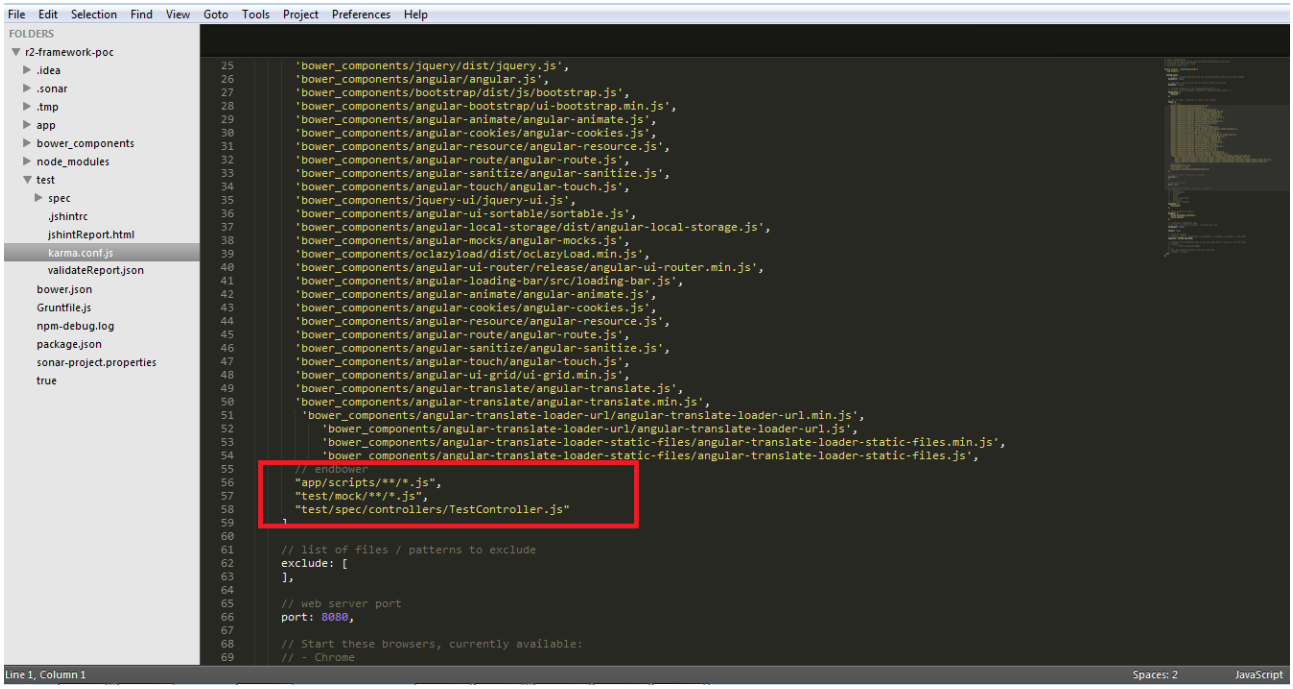
In the above screen shot, last three lines are js files mapped from app and test folder. While running grunt test command, these test files will be tested and the respective source file will be mapped for controller as well as module verification.

And also there are dependency files, which are mapped before we mentioned app and test folder. These dependencies are presents in our angular application modules.

# 5. ADDING DEPENDENCY FILES AND BOWER COMPONENTS

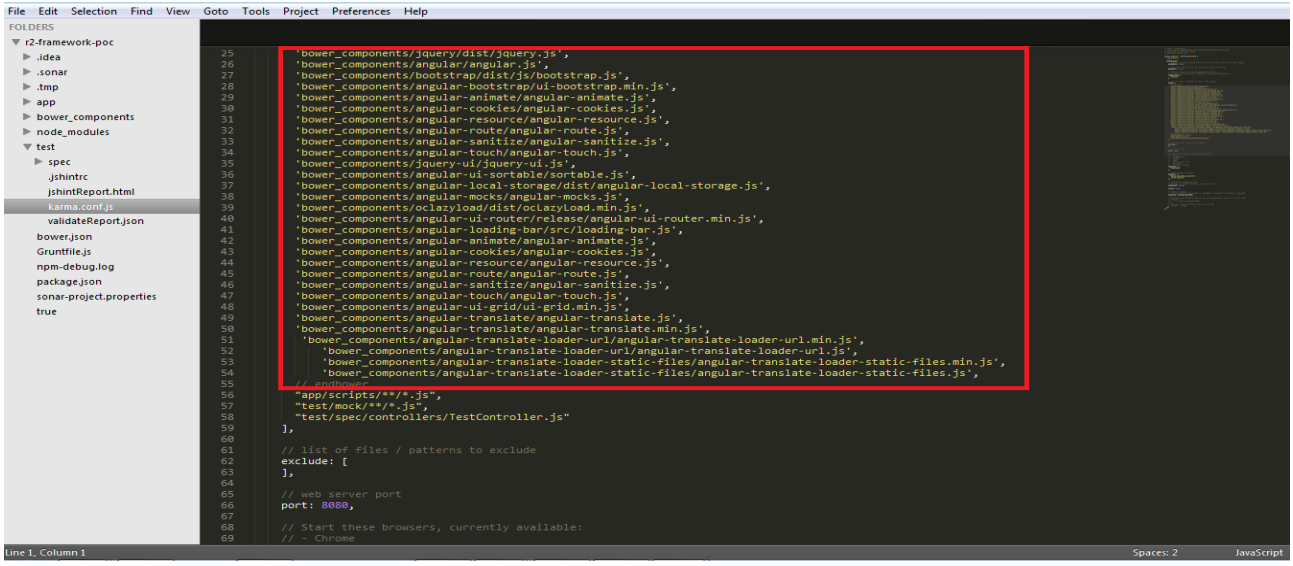
Dependency files and bower components are configured in karma.conf.js file. These dependency and bower files are two types. One is project dependency file and second one is bower components that we used in app modules as sub modules.

**Dependency file sample (Type 1):**




In the above screen shot, you can see that we have mapped our source and test files in karma.conf.js file.

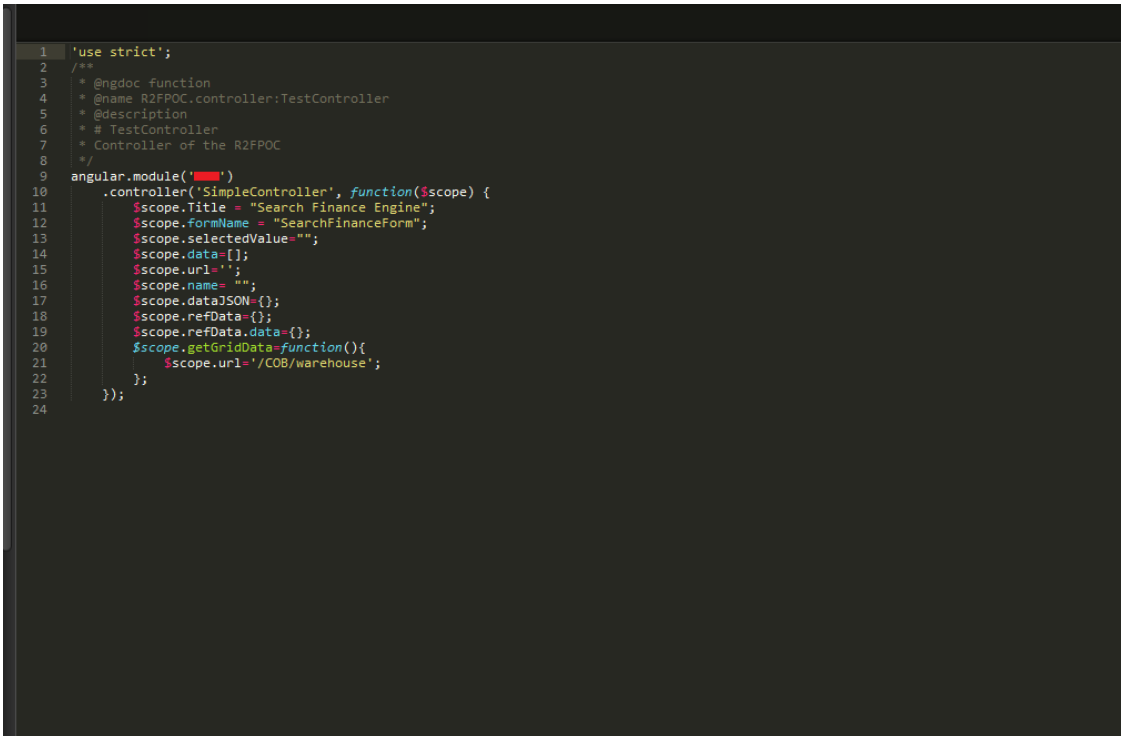
**Dependency file sample (Type 2):**



In above screen shot, highlighted bower dependencies that we are using in our angular JS module as shown below.

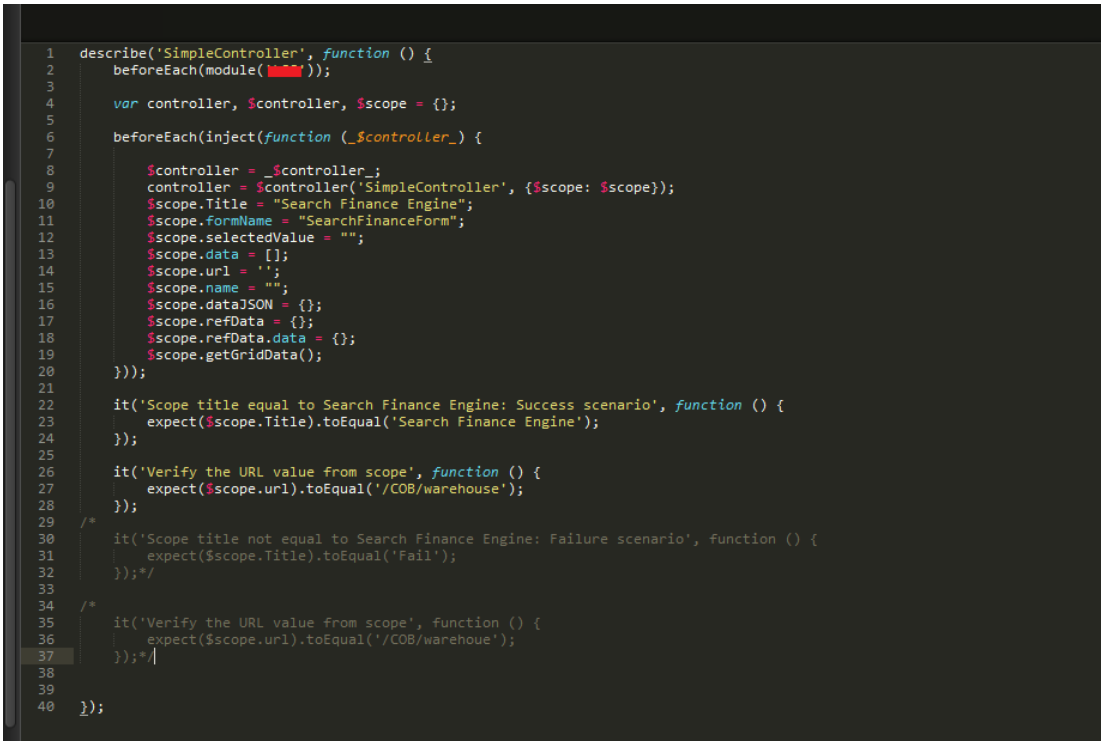
## 6. WRITING TEST FILE USING JASMINE

Look at the screen shots below for the sample source file and test case file.



```
'use strict';
/**
 * @ngdoc function
 * @name R2FPOC.controller:TestController
 * @description
 * # TestController
 * Controller of the R2FPOC
 */
angular.module('[redacted]')
    .controller('SimpleController', function($scope) {
        $scope.Title = "Search Finance Engine";
        $scope.formName = "SearchFinanceForm";
        $scope.selectedValue="";
        $scope.data=[];
        $scope.url='';
        $scope.name= "";
        $scope.dataJSON={};
        $scope.refData={};
        $scope.refData.data={};
        $scope.getGridData=function(){
            $scope.url='/COB/warehouse';
        };
    });
```

**Sample source file**



```
describe('SimpleController', function () {
    beforeEach(module('[redacted]'));

    var controller, $controller, $scope = {};

    beforeEach(inject(function (_$controller_) {

        $controller = _$controller_;
        controller = $controller('SimpleController', {$scope: $scope});
        $scope.Title = "Search Finance Engine";
        $scope.formName = "SearchFinanceForm";
        $scope.selectedValue = "";
        $scope.data = [];
        $scope.url = '';
        $scope.name = "";
        $scope.dataJSON = {};
        $scope.refData = {};
        $scope.refData.data = {};
        $scope.getGridData();
    }));

    it('Scope title equal to Search Finance Engine: Success scenario', function () {
        expect($scope.Title).toEqual('Search Finance Engine');
    });

    it('Verify the URL value from scope', function () {
        expect($scope.url).toEqual('/COB/warehouse');
    });
/*
    it('Scope title not equal to Search Finance Engine: Failure scenario', function () {
        expect($scope.Title).toEqual('Fail');
    });*/

/*
    it('Verify the URL value from scope', function () {
        expect($scope.url).toEqual('/COB/warehoue');
    });*/


});
```

**Sample test file**

**Key concepts in jasmine framework:**

**Describe:**

```
describe('SimpleController', function () { });
```

Describe is the function that will combine all the unit test case into one section with the name of Source Controller name. One test case file may contain more than one describe, each will contain their own test cases writing inside it.

**BeforeEach:**

```
beforeEach(module('PROJECT'));

beforeEach(inject(function (_$controller_) {
    $controller = _$controller_;
    controller = $controller('SimpleController', {$scope: $scope});
    $scope.Title = "Search Finance Engine";
  }));
```

Before each function will load the content inside the function before running each test case. So that the required data and module will be created before running the test case and the populated data will be used by test case block.

**IT:**

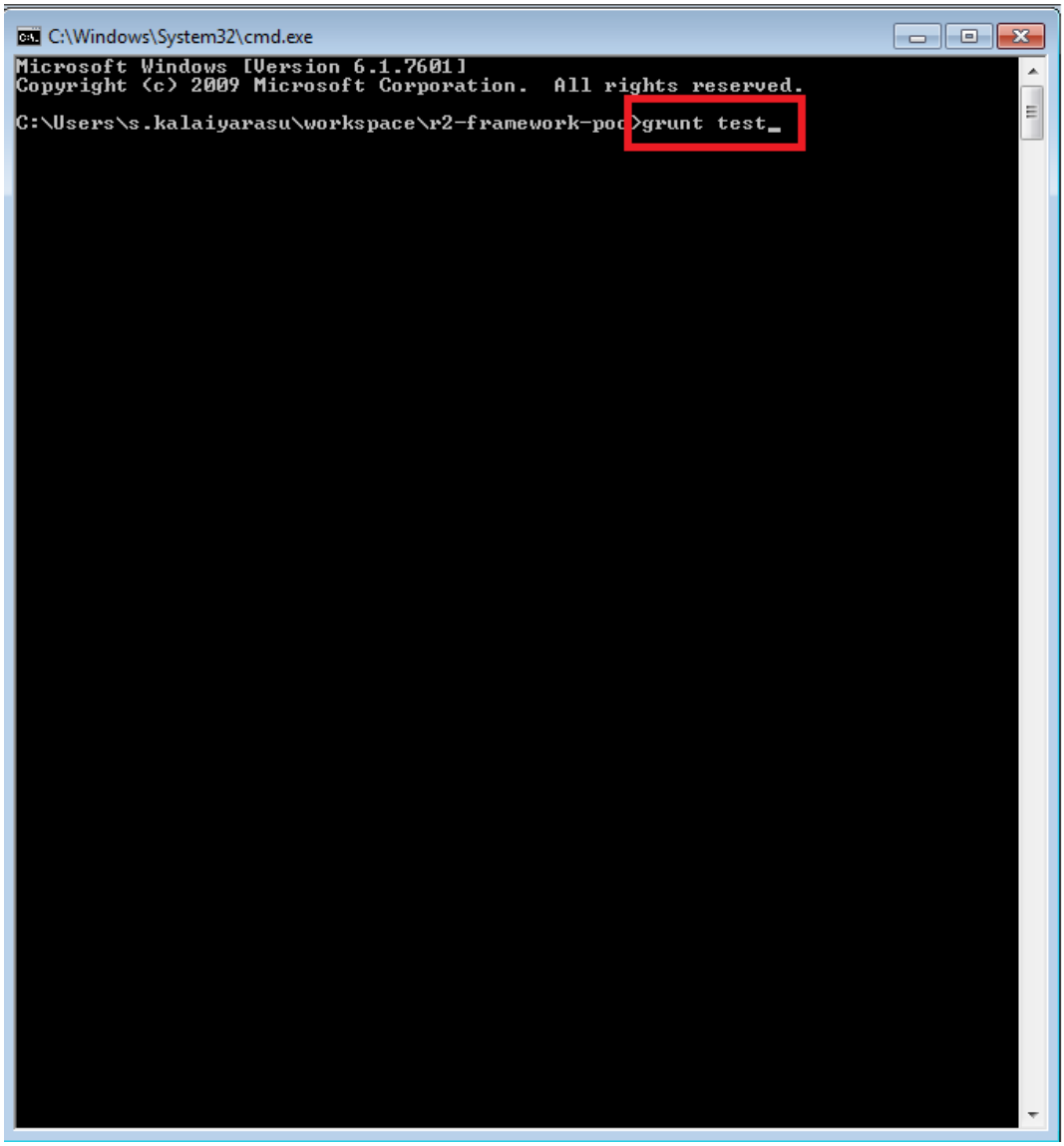
```
it('Scope title equal to Search Finance Engine: Success scenario', function () {
    expect($scope.Title).toEqual('Search Finance Engine');
 });
```

'it' function will act as a test case. Each it function is a separate test case that will be created under describe function. Inside it function we can have some test code as shown above.

Have a look into sample source file and test case file screen shot for better understanding.

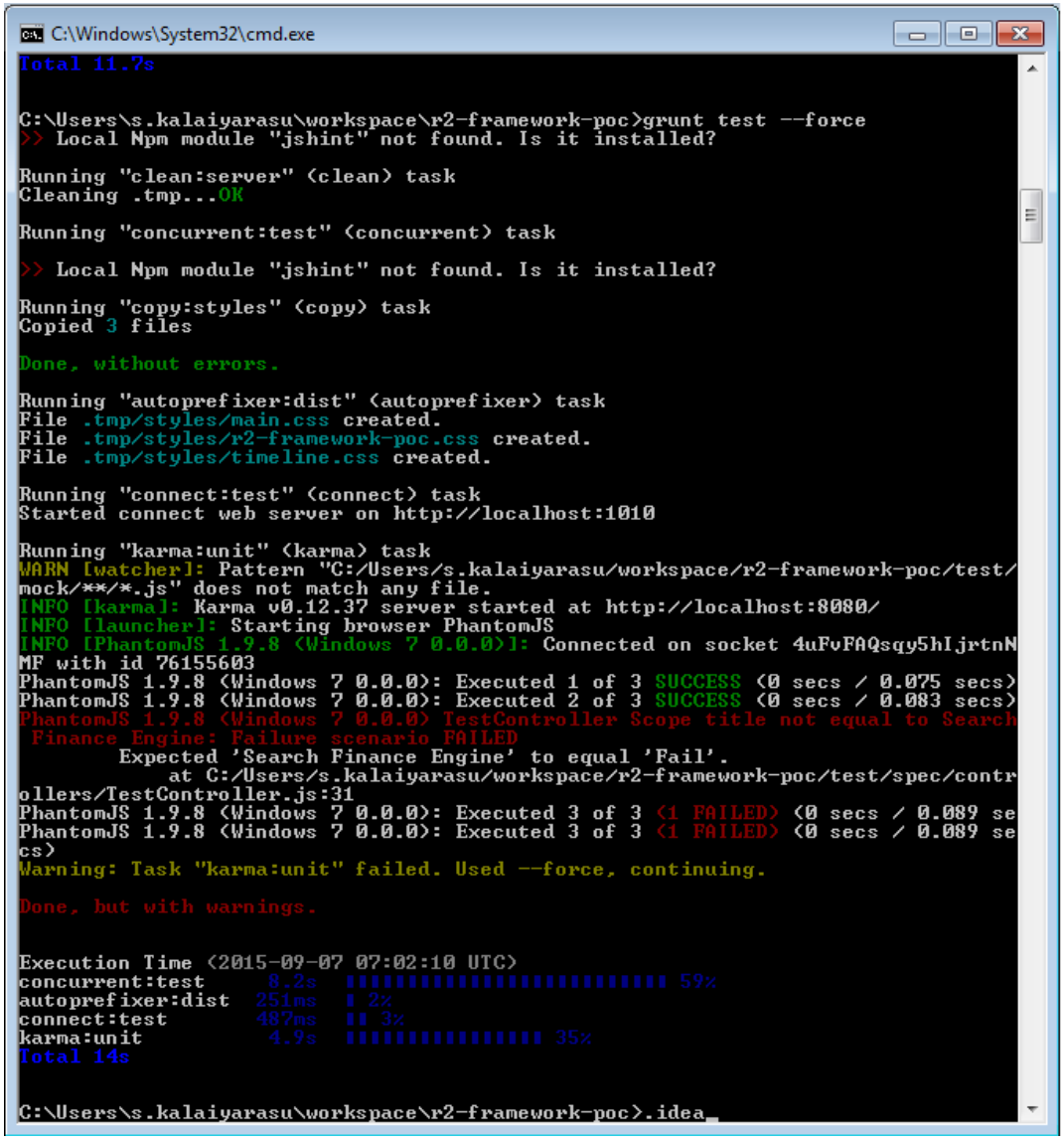
## 7. RUNNING KARMA JASMINE TEST FILE USING KARMA

**grunt test** is the command to run the test file. This command you have to run from your client path command prompt.

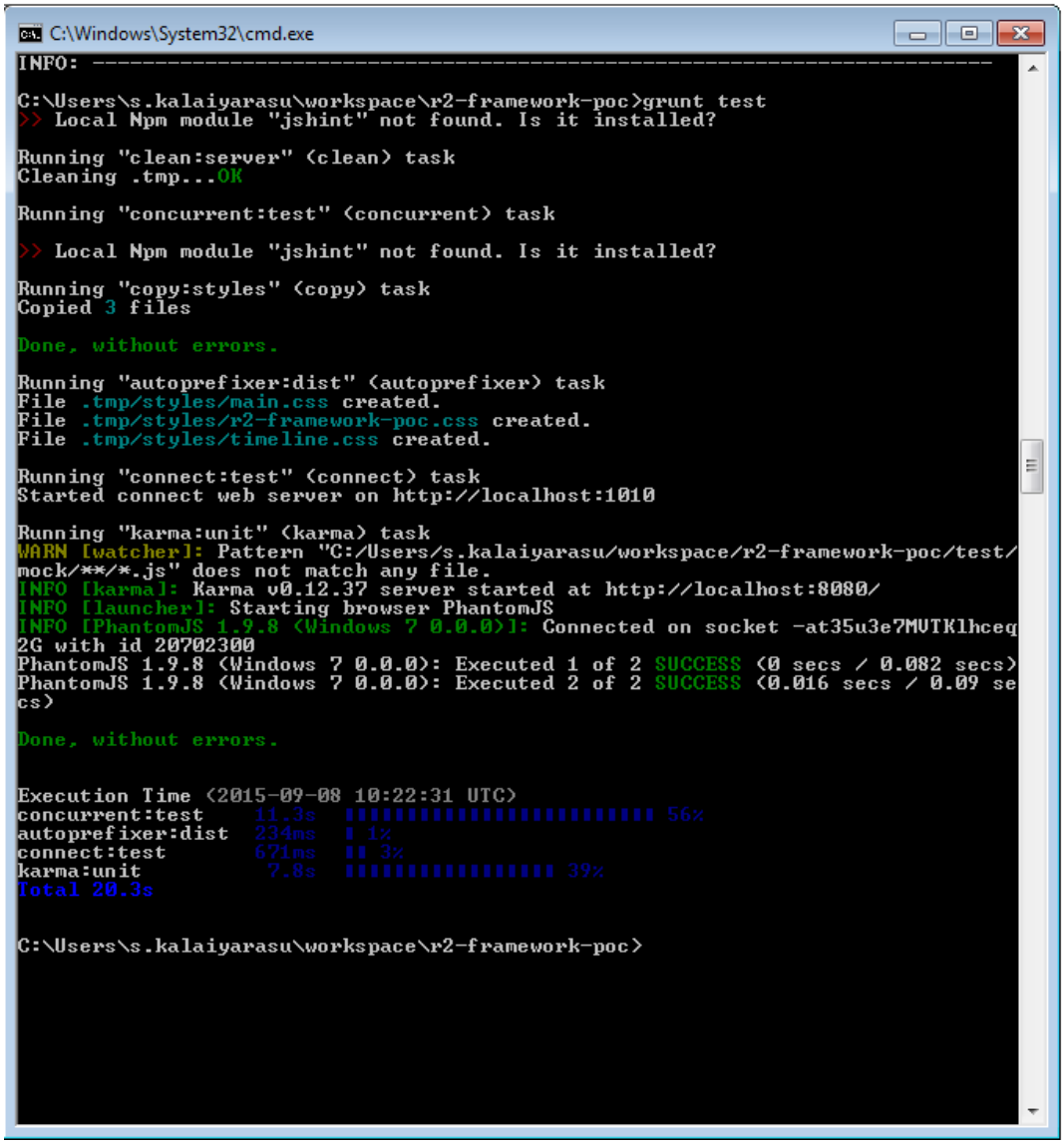



Once the command executed test case successfully, it will show the success and failure count of test case.

**Make sure you are running the test command from your client path**.




In the above test case result, out of three test cases, two are getting passed and one got failure due to expect function not matching with actual result. The failure error will throw with test case description name which is showing in red.




In the above screen shot, executed all the test cases successfully.