**Customer Financials Tracker – Protractor Automation Framework**

**Project Report**

*Submitted in partial fulfilment for the award of the degree*

*of*

**Master of Technology**

***in***

**Information Technology**

*by*

**CHAITRA BABU RAO**

**(15MIN2180)**

*Under the guidance of*

**Prof. Mary Angel Y**

**VIT, Vellore.**



**School of Information Technology and Engineering**

August 2019



**School of Information Technology and Engineering**

**DECLARATION BY THE CANDIDATE**

I hereby declare that the thesis entitled **“Customer Financials Tracker – Protractor Automation Framework”** submitted by me to Vellore Institute of Technology University Vellore, in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Information Technology** is a record of bonafide project work carried out by me under the supervision of **Prof. Mary Angel Y**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**Place**: Bangalore

**Date**: **Signature of the Candidate**



**School of Information Technology and Engineering**

**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled **“Customer Financials Tracker – Protractor Automation Framework”** by CHAITRA BABU RAO **(15MIN2180),** to Vellore Institute of Technology University, Vellore, in partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Information Technology**, is a project bonafide work carried out by him/her under my supervision. The project fulfils the requirement as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this Institute or any other Institute or University.

**Prof. Mary Angel Y**

**Internal Supervisor**

**VIT University**

**Internal Examiner(s) External Examiner(s)**

**Table of Contents**

[**ABSTRACT** 1](#_Toc15746063)

[**LIST OF TABLES** 2](#_Toc15746064)

[**LIST OF FIGURES** 2](#_Toc15746065)

[**TERMS AND ABBREVIATIONS** 3](#_Toc15746066)

[**1.** **INTRODUCTION** 4](#_Toc15746067)

[**1.1** **PROJECT BACKGROUND** 4](#_Toc15746068)

[**1.2 CUSTOMER FINANCIALS TRACKER** 5](#_Toc15746069)

[**2.** **LITERATURE SURVEY** 6](#_Toc15746070)

[**2.1** **EXISTING SYSTEM** 6](#_Toc15746071)

[**2.2** **MANUAL TESTING VERSUS AUTOMATION TESTING** 7](#_Toc15746072)

[**3.** **PROPOSED SYSTEM** 8](#_Toc15746073)

[**3.3.1. FEATURES** 8](#_Toc15746074)

[**3.3.2 ADVANTAGES** 9](#_Toc15746075)

[**3.3.3 DISADVANTGES** 9](#_Toc15746076)

[**3.4 MAJOR DIFFERENCES BETWEEN PROTRACTOR AND SELENIUM WEB DRIVER** 9](#_Toc15746077)

[**3.5 SOFTWARE REQUIREMENTS** 10](#_Toc15746078)

[**4.** **DETAILED DESIGN OF THE SYSTEM** 10](#_Toc15746079)

[**4.1 FRAMEWORK** 10](#_Toc15746080)

[**4.2 WORKING OF THE FRAMEWORK** 12](#_Toc15746081)

[**4.3 MODULES** 13](#_Toc15746082)

[**4.4 FEASIBILITY ASSESSMENT** 17](#_Toc15746083)

[**4.5 SEQUENCE DIAGRAMS** 18](#_Toc15746084)

[**5.** **IMPLEMENTATION OF THE SYSTEM** 21](#_Toc15746085)

[**5.1** **A BRIEF NOTE ON JSON** 27](#_Toc15746086)

[**5.2 TEST CASES** 29](#_Toc15746087)

[**5.2** **SAMPLE SCREENSHOTS** 48](#_Toc15746088)

[**6.** **RESULTS AND DISCUSSIONS** 52](#_Toc15746089)

[**7.** **CONCLUSION AND FUTURE WORK** 63](#_Toc15746090)

# **ABSTRACT**

Customer Financials Tracker is an application that can track the financial reports of a customer related to their properties, investments etc. This is a web based application that allows the user to obtain and save all his/her latest financial details and access the reports from time to time. The Application is built on the NodeJS/AngularJS framework. The best tool to test the application is the Protractor framework that supports both NodeJS and AngularJS. The purpose of using the protractor framework is to test this application end to end (i.e.) right from the login to the subsequent processes that follows post login.

One of the major advantages of the end to end testing is that, during each deployment post the addition of enhancements, manually testing each module would be a rigorous task as it consumes a lot of man power and time. There are chances of missing out on testing certain key features while manually testing the application.

Apart from building the test cases to reduce the manual work, the protractor testing framework will improve the quality of the application by rightly identifying the underlying issues. It can help in reducing the costs and dependency on manual testing during the peak time regression.

# **LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **DESCRIPTION** | **PAGE** |
| TABLE 1.1 | Terms and abbreviations | 3 |
| TABLE 1.2 | Software requirements specification | 9 |
| TABLE 1.3 | Test cases | 40 |

# **LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **TABLE** | **DESCRIPTION** | **PAGE** |
| FIGURE 1.1 | Representative Hub Architecture | 4 |
| FIGURE 1.2 | Protractor framework | 10 |
| FIGURE 1.3 | A Simple working of Protractor | 12 |
| FIGURE 1.4 | A short description | 12 |

# **TERMS AND ABBREVIATIONS**

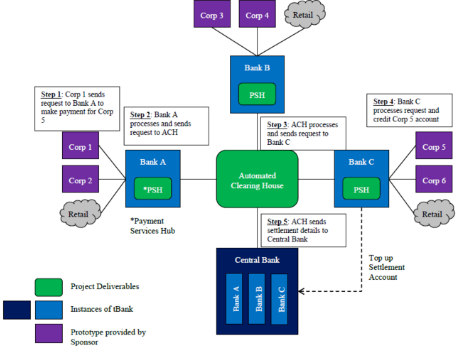
|  |  |
| --- | --- |
| **ABBREVIATION** | **DESCRIPTION** |
| ACH | Automated clearing house |
| PSH | Payments Services HUB |
| BDD | Behavioural Driven Development |

# **INTRODUCTION**

## **PROJECT BACKGROUND**

Every bank has a HUB that is connected to all the commercial systems which holds the record of the user details along with the credit debit information, new facilities and its functioning, the annual reviews, the renewals, the various policies like loan approval details etc.

A representative diagram of a HUB



(FIGURE 1.1)

## **1.2 CUSTOMER FINANCIALS TRACKER**

Customer Financials Tracker is an application that helps the user to understand, process and validate his bank account details and provide the details of the account and the expenses accordingly within a given time period. When the application is developed by a group of individuals, it usually takes a lot of time by the testing team to test each module of the application manually and make note of the working of the modules and then provide a feedback to the developer on the functioning of the application.

This process consumes a lot of time. It could have been saved if the test cases were automated.

The customer financial tracker is helpful in finding the right information of the customer in the right place at the right time without any hassles. There is a database that exists where all the customer details are stored and validated periodically.

The Customer Financials Tracker is not a standalone application. One of the major reliability of this application is on the database and the subsequent calculations that can prove vital to the customer. In order to bring about the right functioning of this application.

\

# **LITERATURE SURVEY**

## **EXISTING SYSTEM**

Manual testing is a process of finding bugs or errors in a developed program or an application in order to validate the processes. The main aim of testing is to reduce the inconvenience for the user and simultaneously ensure the efficiency of the application or a program. In short, it is the process of executing test cases without the help of automation tools.

There are basically two kinds of approaches in testing, an informal approach where it’s a smaller enterprise and the application to be tested is a high level application with minimal backend operations and more of User interface. At a Large enterprise level, the testing can be more aggressive compared to the smaller enterprise since the application is more complex and a systematic procedure is required to ensure the smooth functioning of the application.

The general procedure the large enterprises usually follow as a part of manual testing are:

1. The first level where the licenses are procured and the resources are chosen as per the requirements.
2. Understand the application, work out on the test suite plan and subsequently understand the test cases that are to be added as a part of the suite.
3. Prepare a report where the results can be validated for further processes of development, bug fix and the next level of testing.

Testing as mentioned earlier, has various forms. They can be termed as Black box testing, White Box Testing, Grey Box testing. All the testing forms have a single aim, to ensure the smooth functioning of an application with a minimal inconvenience.

As we all know, the major difference between a black box testing and the white box testing is the access to the code. The black box testing only has a set of test cases that have to be validated while the white box testing has the test cases and as well as the access to the code. Grey Box testing is a combination of both, Black Box and white box testing. This searches for errors if any that can be caused due to the wrong structuring of the applications.

There are other types of testing such as Static Testing and Dynamic Testing where the static testing is carried out before the code deployment while the dynamic testing is carried out post the code deployment. The major con of these kinds of testing is the increase in the cost of the project.

## **MANUAL TESTING VERSUS AUTOMATION TESTING**

Once of the major advantages of automating the process of testing would be the elimination of the manual efforts and reduction in costs. Automation testing also saves a lot of time. When an application has to be manually tested it involves a lot of efforts and is a time consuming process whereas automation testing can run a set of test cases that can provide the results within a short span of time.

Automation testing can be used for large enterprise applications which consists of a large amount of data that has to be tested for defects. The return of investment is higher in the automation testing as compared to manual testing. It also is reliable as compared to manual testing. Automation testing is more valued for Regression Testing and performance testing.

# **PROPOSED SYSTEM**

Even during the recent times, we come across majority of the applications where apart from other forms of testing, manual testing also is still a part of the process although it is time consuming and error prone. Although manual testing is of a great help during small application testing, automation testing can help us ensure minimal inconvenience in terms of working of an application.

The Protractor framework also known as E2E framework supports both NodeJS applications and as well as Angular JS Applications. It was introduced as a part of replacing the existing framework of Angular JS 1.2 framework. The bunch of customizations from Selenium can easily be used to create test cases for applications. It can be a combination of multiple tools. It can work like a solution integrator with multiple technologies in place such as Selenium,NodeJS,webDriver, Jasmine,Mocha and Cucumber.

It works with both Angular and Non-Angular Applications well.

## **3.3.1. FEATURES**

* One of the best features of Protractor is that it speeds up the process of testing E2E.It avoids a lot of Waits and Sleeps
* Protractor is easy to learn if the AngularJS Concepts are known.
* It allows the test cases to be based on Jasmine. The major advantage is that it allows unit and functional test cases.
* It can run on real browsers and as well as headless browsers.

## **3.3.2 ADVANTAGES**

* It is easier to manage the page objects
* Selenium driver feature misses out on default waits while with respect to protractor it is a default feature.
* All the behaviour driven frameworks are supported as a part of protractor
* They have extra locators compared to the Selenium Web Driver.
* Since it runs on the Selenium Webdriver, all the features are supported.

## **3.3.3 DISADVANTGES**

* It is compatible with Chrome browser while other browser supports are negligent.
* It can support only javacript
* Protractor supports only Robot Class

**3.4 MAJOR DIFFERENCES BETWEEN PROTRACTOR AND SELENIUM WEB DRIVER**

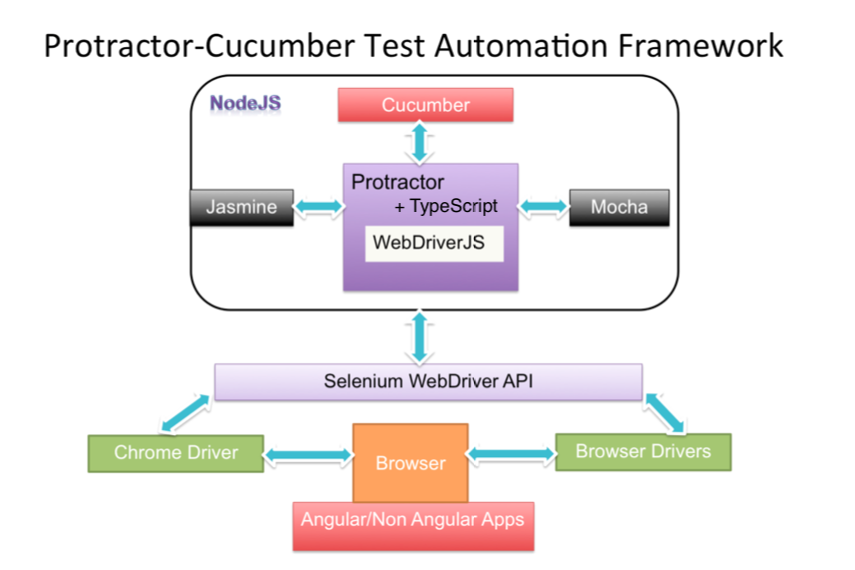
* Protractor acts as a syntax wrapper for the selenium automation, with built in angular friendly methods.
* The major choice is Protractor and selenium iis purely based on the syntax and language preference,
* Protractor has a better documentation and lesser code as compared to Selenium, (i.e) the coding in Selenium takes a lot more time as compared to the Protractor.

## **3.5 SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| **Tools** | NodeJS, Protractor 5,Angular JS and Jasmine Framework |
| **Language** | Javascript |
| **OS** | Windows7 and Above |
| **Browsers** | Google Chrome, Mozilla Firefox |

# **DETAILED DESIGN OF THE SYSTEM**

## **4.1 FRAMEWORK**



(FIGURE 1.2)

**DESCRIPTION**

**JASMINE**

* It is a default framework when the protractor is installed.
* It is an easy-to-read syntax
* It is influenced by other unit testing frameworks such as JSpec, JSSpec.

**MOCHA**

* A test framework that runs on Node.JS
* It makes it easier for asynchronous testing and a BDD interface can be used.
* It is an Open source framework

**CUCUMBER**

* It supports the behavioural driven development.
* It can be used to write the acceptance tests
* The code can be written in a simple format which is understandable

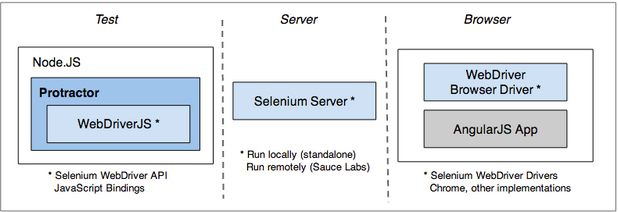
**CHROME DRIVER**

* It is a separate executable web driver
* It is used to control chrome

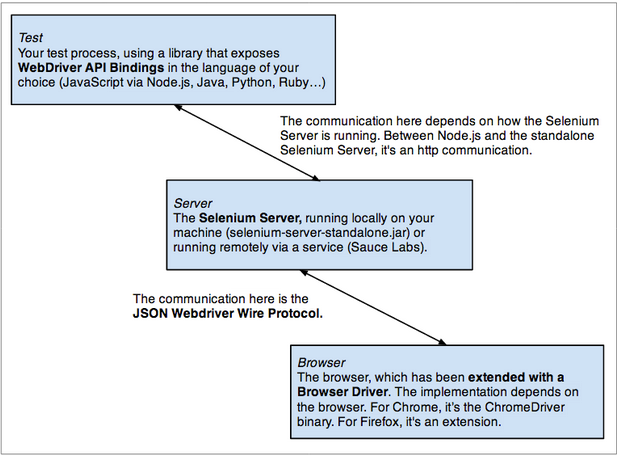
**SELENIUM WEBDRIVER API**

* Tool for automating web application testing
* Works as a friendly API

## **4.2 WORKING OF THE FRAMEWORK**



(FIGURE 1.3)



(FIGURE 1.4)

## **4.3 MODULES**

The major modules in the project are as follows:

* 1. Spec Files
  2. Support Files
  3. TestReports
  4. Conf.js File

1. **Spec Files**:

* Spec files is one of the two important files required to run the protractor
* This contains the logic and the locators to interact with the application
* In a Spec File we describe the components of the application that can be a class or a function
* This file works with the configuration file

The below code is one of the Spec Files used for testing

|  |
| --- |
| describe('Protractor App Account Search', function() {  it('Account Summary Search', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });  browser.get('https://secure.budgettracker.com/login\_secure.php');  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  browser.sleep(5000);  element(by.xpath('//\*[@id="apps\_titlebar1Id"]/table/tbody/tr/td[2]/a')).click();  browser.sleep(10000);  expect(element(by.xpath('//\*[@id="acctsumToolbarID"]/table[3]/tbody/tr[1]/td/span')).getText()).toEqual('$561.67');  }); }); |

1. **Support Files** :

Support files are a group of files that are required to support the framework. It is to ensure the smooth functioning of the framework.

1. **Test Reports**

One of the major advantages of using the Protractor framework is its able to report the success and failure rate of the test cases along with the screenshots making it a very easy task to report the outcome and provide the feedback.

The following code is an example for the reports can be obtained

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',  specs: ['AccountSummary.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

1. **Configuration File**

The configuration file helps the protractor to locate where the test files have been placed and to connect to the Selenium Server. Chrome is the default browser for the Protractor.

It gives an explanation for all the Protracotr configuration options.The deafulat framework used in the protractor is the jasmine framework.

Below are the sample codes for the Configuration file

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['logintest.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['language.js'] ,  jasmineNodeOpts:{  defaultTimeOutInterval:50000  },  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());    }  }; |

## **4.4 FEASIBILITY ASSESSMENT**

Every project likewise, are assessed for the feasibility and ROI. One of the major objectives of this project is to reduce the manual efforts and the time taken to complete the E2E testing with results. We have already discussed about the advantages and disadvantages in the previous section.

In this section, we can clearly state that the protractor framework clearly holds the upperhand since there is a minimal effort required by the testers in order to test an application, be it a small enterprise or a large enterprise.Although the application is developed at a market cost the testing process can always be of minimal cost based on thetype of testing methology that has been adopted.There is a very less amount of manual intervention that is required in order to execute the test cases. The framework can be seamlessly integrated with the browser applications that can run via the config file. The automation process certainly involves initiating a plan, devising the plan, continuous monitoring, cases execution, review and enhancements.

As a part of above mentioned processes, the workflow creation during the planning stage is also a major decider in terms of RoI. If the workflow is unstructured, a test case might be overseen due to improper format.

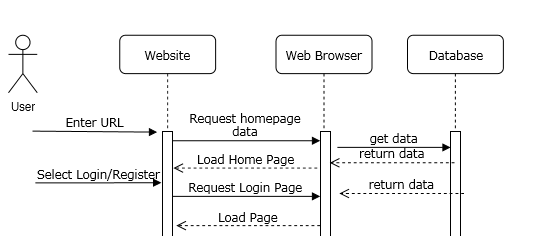
While we concentrate on the present scenarios of the project, a future enhancement also has to be a valid part of RoI and is rightly so, since prtracotr can be a tool of great help to testers.The quality of automation provided by the Protractor frameowkr also is of utmost importance. If the Application can be tested successfully from E2E with a minimal human intervention it is a clear win there.

## **4.5 SEQUENCE DIAGRAMS**

A sequence diagram usually represents the object interactions in a particular time sequence. It is one of the behavioural diagrams that captures the interactions that take place as a part of collaborations which can be a use case or also an operation. It basically represents the high level interactions between the systems or sub-systems. The objects are placed horizontally while the vertical line represents time.

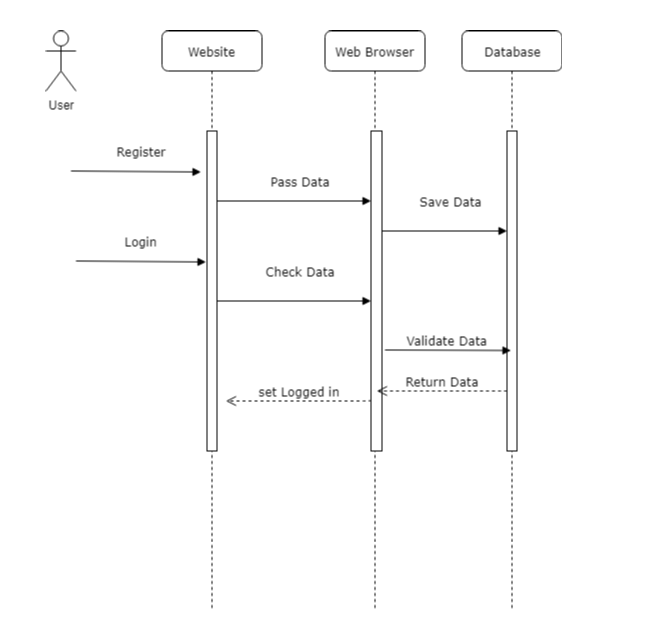
1. The Home Page Sequence Diagram For Login

Here, the user-the actor enters the URL and requests the homepage data. The browser connects to the database and fetches the details and loads the homepage.Once the homepage has been loaded to the browser, the User has to enter the credentials which are then verified in the backend.Once it has been validated, the user will login to see the homescreen with the details of his account and the summaries of the transactions.



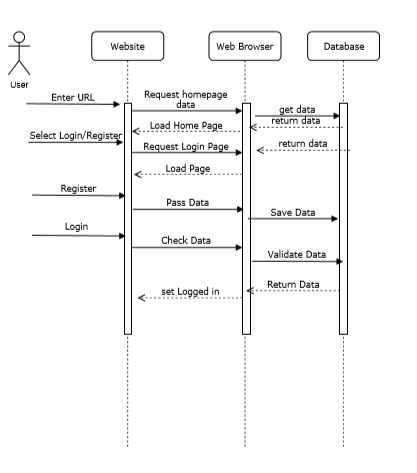
1. The Home Page Sequence Diagram For Sign-Up

If the user is new to the site, the user can register through the sign up option provided on the homepage and enter the details like his email address and set up a password of his choice. If the email address is unique and not present in the system, the user will be created with the set of new credentials.

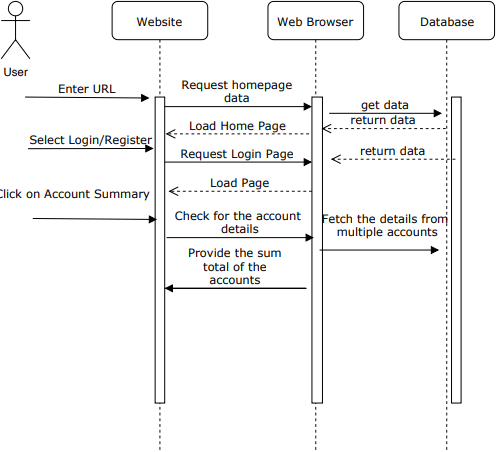


1. The Home Screen Page

This is the summary of the login and sign up screen as a single page with multiple interaction with the web browser and database.



1. **The Account Summary**



## **IMPLEMENTATION OF THE SYSTEM**

The initial phase of this project is to choose an application that can be tested and noted down for the success and failure results. Once the application has been chosen, the next step is to install Node.JS.

NodeJS is an open server environment that allows us to run the java script on the server. It is built on Google Chrome's Java script engine. It can be used for scalable applications.

Once the NodeJS has been installed, the npm package that comes with the Node JS can be used to install the prtracotr framework. This has to be done from the command prompt. The below command will install the Protractor globally.

Npm install –g protractor

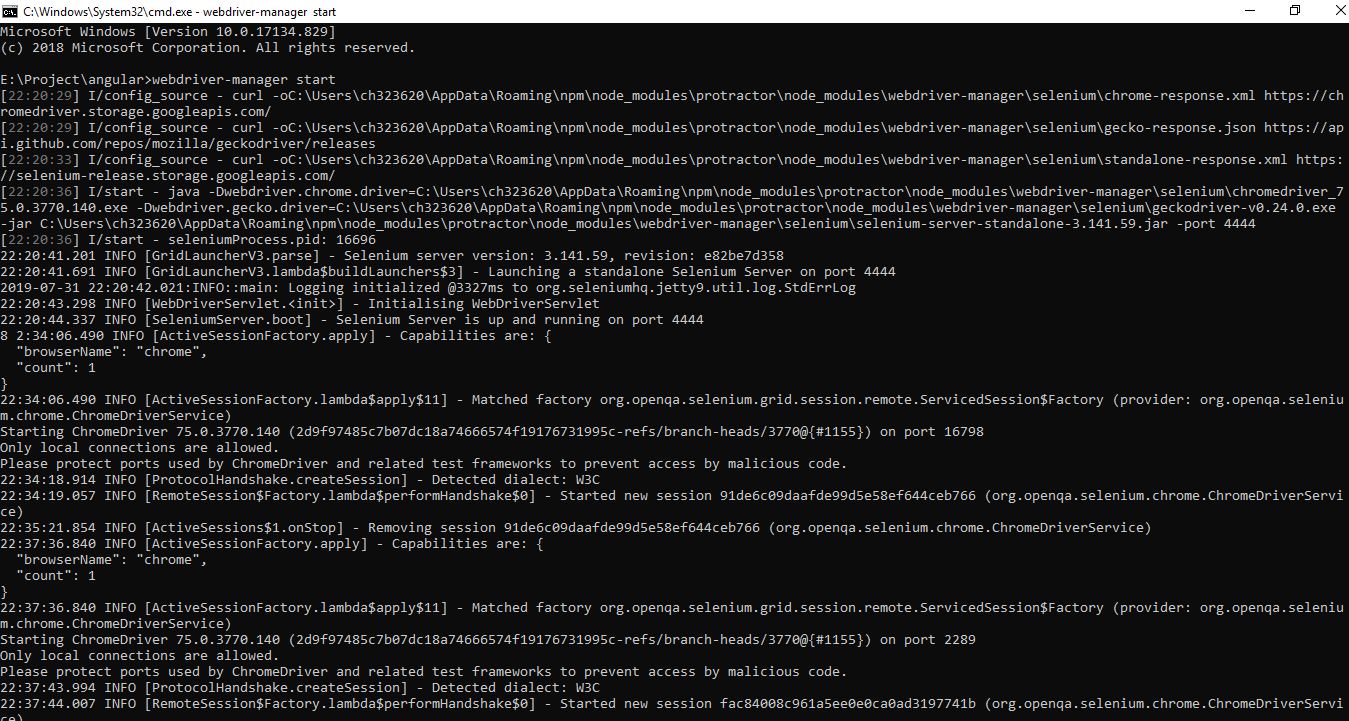
-g installs the tool globally.It will be installed along with the Protractor API and a default selenium server.This is an added advantage because we do not have to start a standalone server.

Once the protractor has been installed, the selenium server and the Chrome Driver have to be installed. This can be done by running the below command in the command prompt.

Webdriver –managerupdate

We can start using protractor post the installation. The code can be written in any IDE /editor. Eclipse IE and Atom are the most popular ones.

Web Driver Start

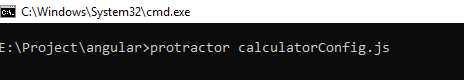


There are two files that are required to run the test case.

1. Configuration File

As explained in the previous section, the configuration file is necessary for running the test cases. This acts as a locator to obtain the details of the test cases that have to be run for the validation and as well as the connector to the Selenium server etc.

Once the configuration file is ran through the command prompt, it fetches the Javascript that has to be run as a test case. Below is the snapshot from the command prompt where the configuration file is being run.



The calculatorConfig executed. We can see that the spec details contains the file name Feedback.js, which holds the code for running the test case.

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['calculator.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Code Breakdown:

* The test case, while being run can be added as a part of a report,can take screenshots and provide a success failure report of the application test cases.
* In the specs section, the path of the spec file has been specified.
* As the code suggests, the framework used here is Jasmine.

1. Spec File

Once the configuration file has been exexuted, the sepc file code is checked for the case.The function, time interval, the variables present on the screen, popup alerts if any, are tested for errors.

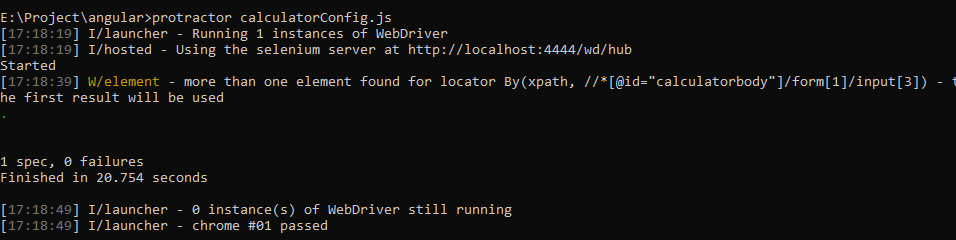
|  |
| --- |
| describe('Protractor App Account Search', function() {  it('Budget Calculator', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 1000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });    browser.get('https://secure.budgettracker.com/login.php?sp=nouser');  //Click on Calculator link  element(by.partialLinkText('Calculators')).click();  //Paycheck Net Amount: field  element(by.name('netincome')).sendKeys(5000);  //Other Income: field  element(by.name('otherincome')).sendKeys(100);  //Click on Update income button  element(by.xpath('//\*[@id="calculatorbody"]/form[1]/input[3]')).click();  //login -end  browser.sleep(9000);  browser.pause();  //Total Income- botton of page  expect(element(by.id('totalincome')).getText()).toEqual('5000');  });  }); |

Code Breakdown:

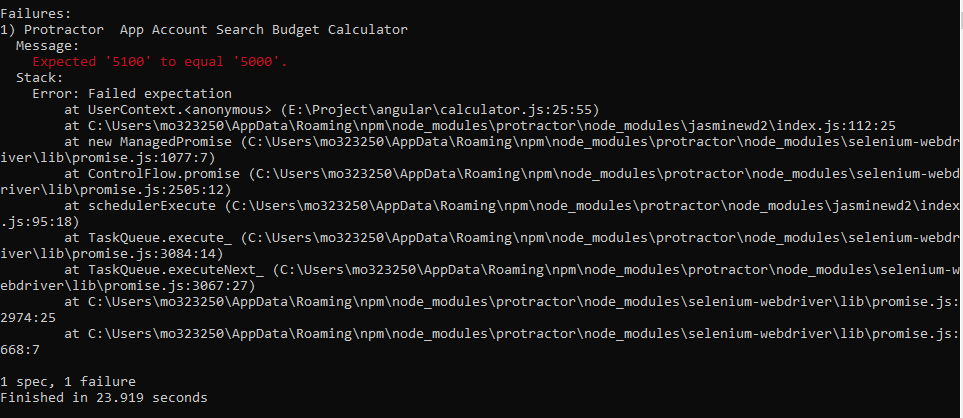
* The browser level commands are handled by the browser a global created by the Protractor.
* The "it" and "describe" are the part of syntax of the jasmine framework.
* By default the protractor works with the angular framework, the code has to be specified if the code is non angular in the browser.synchronizartion tag.

If there are any errors/files missing during the running of the case, although the test case will be successful, the error will be thrown as a part of the result.

The sample result run from the command prompt with the result is shown in the next page.



Sample screenshot for the failure rate.



One of the major advantages of the protractor framework is the screenshots folder that is created temporarily with all the details about the report and screenshots along with a JSON file for every test case, that holds the details of the test case, the result and the screenshot name.

In the code, we do have an option where we can capture screenshots only for failed test cases, skipped test cases etc.

## **A BRIEF NOTE ON JSON**

JSON- Java Script Object Notation, is an open standard file format that is user friendly. It has the data representation in the form of attribute-value pairs.It is independent of the language and a stateless real time server to browser communication protocol. They have their own datatypes.

The JSON File looks like below for a successful test case,

|  |
| --- |
| {"description":"Budget Calculator|Protractor Budget Calculator",  "passed":true,  "pending":false,  "sessionId":"e50fb97efb307e2daacbbeb20964b255",  "instanceId":23784,  "browser":{"name":"chrome"},  "message":"Passed.",  "trace":"","browserLogs":[],  "screenShotFile":"00830011-005f-0038-0013-0092007d00a9.png",  "timestamp":1564490454081,  "duration":24404  } |

For a test case that has a timeout error, the JSON File looks like,

|  |
| --- |
| {"description":"Add Account Details|Protractor App ADD ACCOUNT",  "passed":false,"pending":false,  "sessionId":"46a7b20cb7f7d2564918055544765e08",  "instanceId":18384,  "browser":{"name":"chrome"},  "message":["Error: Timeout - Async callback was not invoked within timeout specified by jasmine.DEFAULT\_TIMEOUT\_INTERVAL."],  "trace":["Error: Timeout - Async callback was not invoked within timeout specified by jasmine.DEFAULT\_TIMEOUT\_INTERVAL.\n  at Timeout.\_onTimeout (C:\\Users\\ch323620\\AppData\\Roaming\\npm\\node\_modules\\protractor\\node\_modules\\jasmine-core\\lib\\jasmine-core\\jasmine.js:4281:23)\n at ontimeout (timers.js:424:11)\n at tryOnTimeout (timers.js:288:5)\n at listOnTimeout (timers.js:251:5)\n at Timer.processTimers (timers.js:211:10)"],  "browserLogs":[{"level":"WARNING","message":"https://secure.budgettracker.com/inc/jquery-1.6.1.min.js 17 'webkitRequestAnimationFrame' is vendor-specific. Please use the standard 'requestAnimationFrame' instead.","timestamp":1564631328809,"type":""}],"screenShotFile":"00740082-0074-009d-0001-003c006d0040.png","timestamp":1564631312186,"duration":30386 } |

## **5.2 TEST CASES**

1. Login

Configuration File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['logintest.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File

|  |
| --- |
| describe('Protractor App', function() {  it('Login Function', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });  browser.get('https://secure.budgettracker.com/login\_secure.php');  element(by.name('username')).sendKeys('Email ID');  element(by.name('password')).sendKeys('Password');  element(by.className('button')).click();  browser.sleep(5000);  expect(element(by.className('whiteTxt')).getText()).  toEqual('Logged in as:');  });  }); |

1. Feedback

Configuration File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['Feedback.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['Feedback.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

1. Calculator

Configuration File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['calculator.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File

|  |
| --- |
| describe('Protractor App Account Search', function() {  it('Budget Calculator', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 1000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });    browser.get('https://secure.budgettracker.com/login.php?sp=nouser');  //Click on Calculator link  element(by.partialLinkText('Calculators')).click();  //Paycheck Net Amount: field  element(by.name('netincome')).sendKeys(5000);  //Other Income: field  element(by.name('otherincome')).sendKeys(100);  //Click on Update income button  element(by.xpath('//\*[@id="calculatorbody"]/form[1]/input[3]')).click();  //login -end  browser.sleep(9000);  browser.pause();  //Total Income- botton of page  expect(element(by.id('totalincome')).getText()).toEqual('5000');  });  }); |

1. Mortgage

Configuration File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['mortageCalculator.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File:

|  |
| --- |
| describe('Protractor Budget Calculator', function () {  it('Budget Calculator', function () {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });  browser.get('https://secure.budgettracker.com/login.php?sp=nouser');  //Click on Calculator link  element(by.partialLinkText('Calculators')).click();  //Dropdown - 3 option mortage calculator  element(by.xpath('//\*[@id="calculatorsId"]/option[3]')).click();  //Mortgage Amount:  element(by.id('mamount')).sendKeys(100);  //Mortgage Term in Years: field  element(by.id('myears')).sendKeys(100);  //Interest Rate per Year: field  element(by.id('minterest')).sendKeys(1);  //Click on Calculator mortage button  element(by.xpath('//\*[@id="calculatorbody"]/form/input[4]')).click();  //login -end  browser.sleep(8000);  //Total Income- botton of page  expect(element(by.id('totalMortageId')).getText()).toEqual('Your monthly mortgage payment will be: $8.33');  });  }); |

1. Account Summary

Configuration file

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['AccountSummary.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File

|  |
| --- |
| describe('Protractor App Account Search', function() {  it('Account Summary Search', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });    browser.get('https://secure.budgettracker.com/login\_secure.php');  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  browser.sleep(5000);  element(by.xpath('//\*[@id="apps\_titlebar1Id"]/table/tbody/tr/td[2]/a')).click();  browser.sleep(10000);  expect(element(by.xpath('//\*[@id="accountbody"]/table/tbody/tr/td[2]/div/div/span')).getText()).toEqual('$295.37');  });  }); |

1. Language

Config File

|  |
| --- |
| describe ("test budget tracker", function(){  it ("to test login", function(){  browser.ignoreSynchronization = true;  browser.get("https://secure.budgettracker.com/login\_secure.php");  element(by.name("username")).sendKeys("marianntsh@gmail.com");  element(by.name("password")).sendKeys("rosh@2210");  element(by.buttonText("Login")).click();  browser.sleep(10000);  element(by.linkText("Spanish")).click();  browser.sleep(10000);  expect(element(by.className("whiteTxt")).getText()).toEqual("Estás conectado como:");  browser.sleep(5000);  });  }); |

Spec File

|  |
| --- |
| describe ("test budget tracker", function(){  it ("to test login", function(){  browser.ignoreSynchronization = true;  browser.get("https://secure.budgettracker.com/login\_secure.php");  element(by.name("username")).sendKeys("marianntsh@gmail.com");  element(by.name("password")).sendKeys("rosh@2210");  element(by.buttonText("Login")).click();  browser.sleep(10000);  element(by.linkText("Spanish")).click();  browser.sleep(10000);  expect(element(by.className("whiteTxt")).getText()).toEqual("Estás conectado como:");  browser.sleep(10000);  });  }); |

1. New account

Config File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['newAccount.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec

|  |
| --- |
| describe('Protractor App ADD ACCOUNT', function() {  it('Add Account Details', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });    browser.get('https://secure.budgettracker.com/login\_secure.php');  //login -start  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  //login -end  browser.sleep(5000);  //click on New Account button  element(by.xpath('//\*[@id="subnavnewentryId"]/a')).click();  browser.sleep(5000);  //Account name filed  element(by.xpath('//\*[@id="accountNameId"]')).sendKeys("TESTAMOUNT")  //Current balance filed  element(by.xpath('//\*[@id="newaccountID2"]/tbody/tr[2]/td[2]/input')).sendKeys(10);  //Account NUmber filed  element(by.xpath('//\*[@id="accountNumberId"]')).sendKeys(1234568);  //Account Type field -dropdown - 4 options  element(by.xpath(`//\*[@id="newaccountID2"]/tbody/tr[4]/td[2]/select[1]/option[4]`)).click();  //Current balance field -dropdown - 3 option  element(by.xpath('//\*[@id="acctcurrencyidnew"]/option[3]')).click();  browser.sleep(5000);  //Exchnage rate text field  element(by.name('exchangerate')).sendKeys(20);  //Xpath of ADD account button  element(by.xpath('//\*[@id="newaccountID2"]/tbody/tr[9]/td/input[1]')).click();  browser.sleep(3000);  //Total amount  expect(element(by.xpath('//\*[@id="accountupdate\_div"]')).getText()).toEqual('Success! Your account has been saved.');  });  }); |

1. Delete

Config File

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['delete.js'] ,  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec File

|  |
| --- |
| describe('Protractor App Account Search', function() {  it('Account Summary Search', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });    browser.get('https://secure.budgettracker.com/login\_secure.php');  //login -start  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  //login -end  browser.sleep(5000);  //click on first check box  element(by.name('account\_1')).click();  //click on delete button  browser.sleep(5000);    element(by.className('redButton button')).click();  let ale = browser.switchTo().alert();  ale.accept();  //Check the available balance  browser.sleep(5000);  browser.sleep(5000);  expect(element(by.xpath('//\*[@id="acctsumToolbarID"]/table[3]/tbody/tr[1]/td/span')).getText()).toEqual('$561.67');  });  }); |

1. Search

Spec File

|  |
| --- |
| describe('Protractor App', function() {  it('should Search with Rent Tracke keyword in Apps Tab', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 10000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });  browser.get('https://secure.budgettracker.com/login\_secure.php');  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  browser.sleep(5000);  element(by.partialLinkText('APPS')).click();  browser.sleep(5000);  element(by.id('searchboxID')).sendKeys('Rent Tracke');  element(by.xpath('//\*[@id="Common\_Main"]/table/tbody/tr[2]/td/table/tbody/tr[1]/td/table/tbody/tr/td[3]/form/input[2]')).click();  expect(element(by.partialLinkText('Rent Tracker')).getText()).toEqual('Rent Tracker');  });  }); |

10. Reporting

Conf.js

|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',  specs: ['Report.js'] ,  jasmineNodeOpts: {  defaultTimeoutInterval: 100000  },  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

Spec.js

|  |
| --- |
| describe('Protractor App', function() {  it('Login Function', function() {  browser.ignoreSynchronization = true  beforeEach(function (done) {  window.jasmine.DEFAULT\_TIMEOUT\_INTERVAL = 50000;  setTimeout(function () {  console.log('inside timeout');  done();  }, 5000);  });  browser.get('https://secure.budgettracker.com/login\_secure.php');  element(by.name('username')).sendKeys('marianntsh@gmail.com');  element(by.name('password')).sendKeys('rosh@2210');  element(by.className('button')).click();  browser.sleep(5000);  element(by.linkText("Reporting...")).click();  browser.sleep(5000);  element(by.linkText("Reports & Graphs")).click() });  }); |

11. Transaction

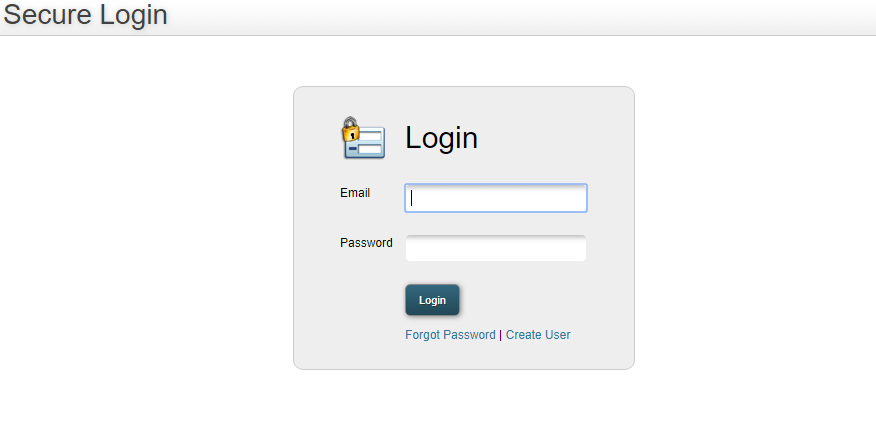
Conf File

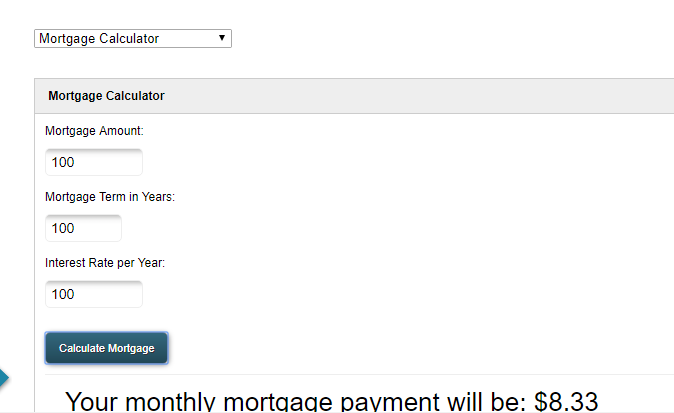
|  |
| --- |
| var HtmlReporter = require('protractor-beautiful-reporter');  exports.config = {  seleniumAddress: 'http://localhost:4444/wd/hub',    specs: ['transaction.js'] ,    jasmineNodeOpts: {  defaultTimeoutInterval: 100000  },  onPrepare: function() {  jasmine.getEnv().addReporter(new HtmlReporter({  baseDirectory: 'tmp/screenshots'  }).getJasmine2Reporter());  }  }; |

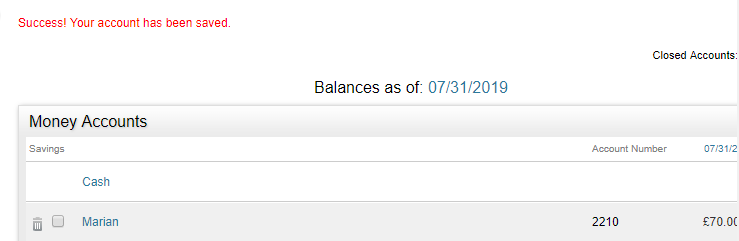
Spec File.

|  |
| --- |
| describe ("test budget tracker", function(){  it ("to test login", function(){  browser.ignoreSynchronization = true;  browser.get("https://secure.budgettracker.com/login\_secure.php");  element(by.name("username")).sendKeys("marianntsh@gmail.com");  element(by.name("password")).sendKeys("rosh@2210");  element(by.buttonText("Login")).click();  browser.sleep(10000);  /\*element(by.linkText("Marian")).click();  var bal1= element(by.name("current\_balance"));//check the current balance of Marian  browser.sleep(5000);  element(by.linkText("My Accounts")).click();  browser.sleep(10000);  element(by.linkText("Moses")).click();  var bal2= element(by.name("current\_balance"));//check the current balance of Moses  browser.sleep(000);\*/  element(by.linkText("My Transactions")).click();  browser.sleep(10000);  element(by.linkText("Add a Transaction")).click();  browser.sleep(5000);  element(by.name("tpaytonew")).sendKeys("Marian to Moses Transfer");  element(by.name("account")).$('[value="633443"]').click();//Set from account to Moses  browser.sleep(5000);  //element(by.id("transferToId")).$('[value="633368"]').click();//Set to account to Marian  element(by.name("tamount")).sendKeys("10");  element(by.name("enter")).click();  browser.sleep(5000);  });  }); |

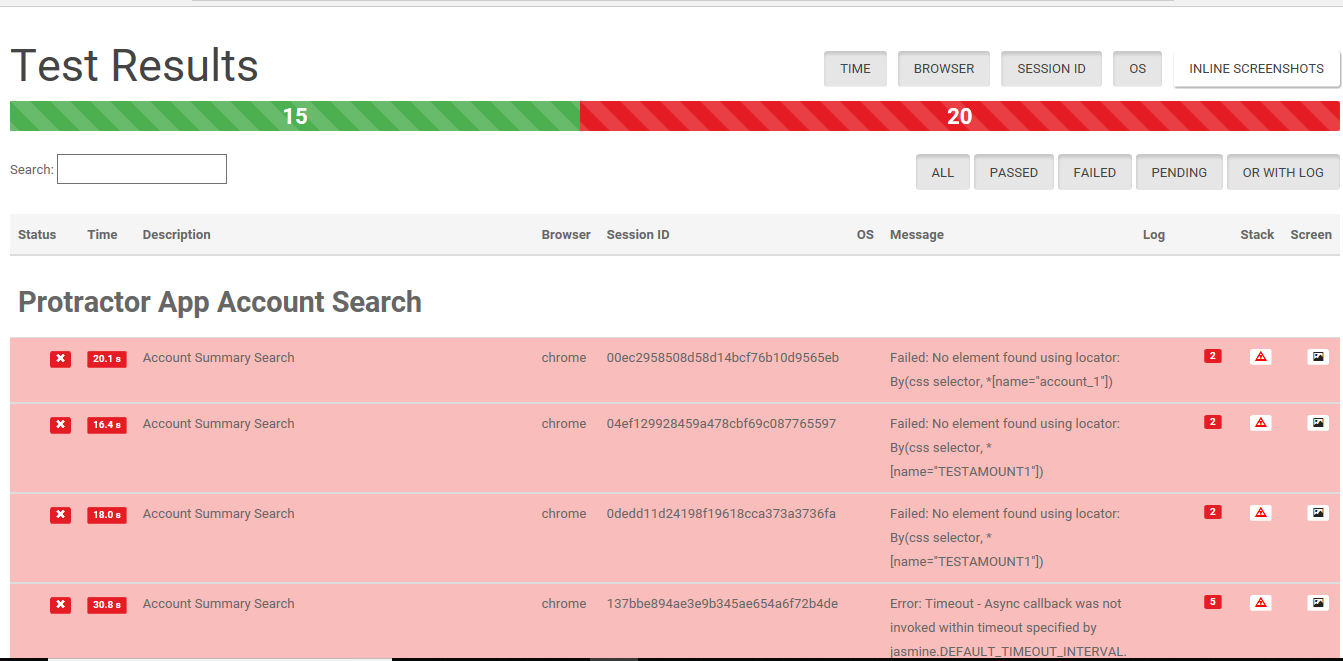
## **SAMPLE SCREENSHOTS**

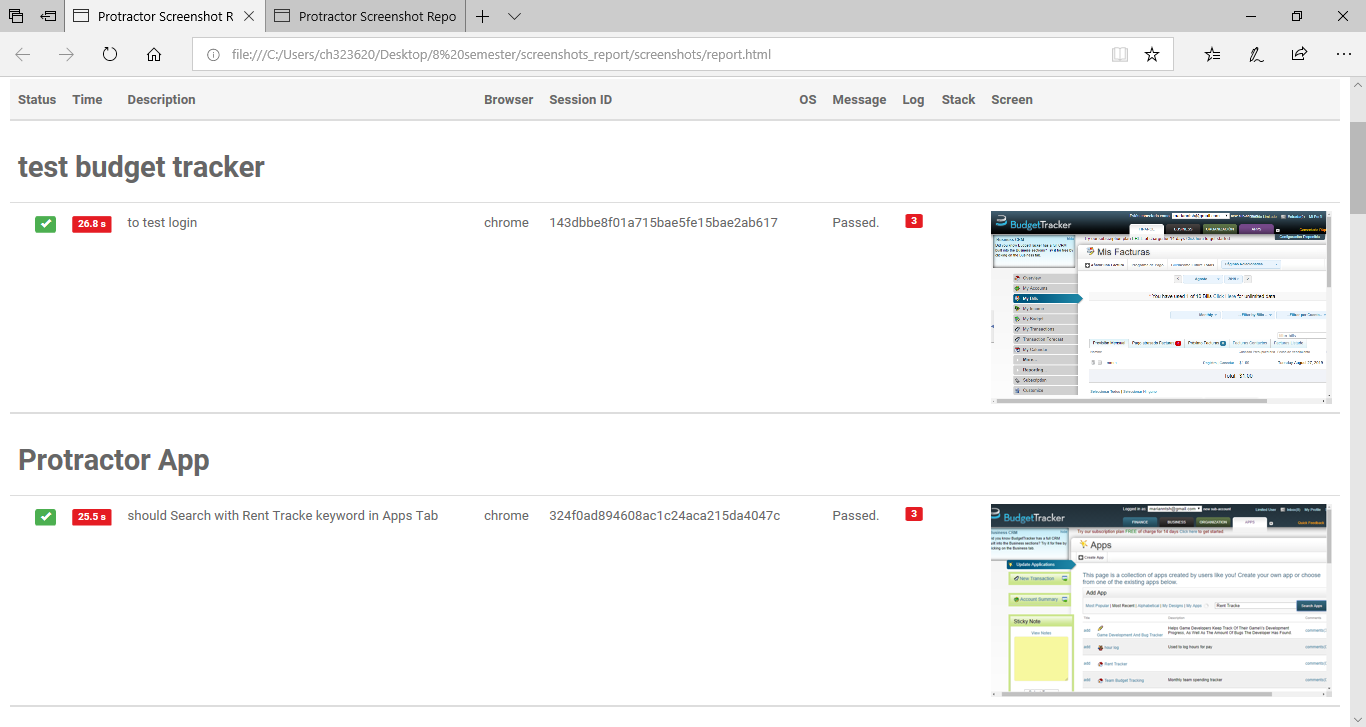


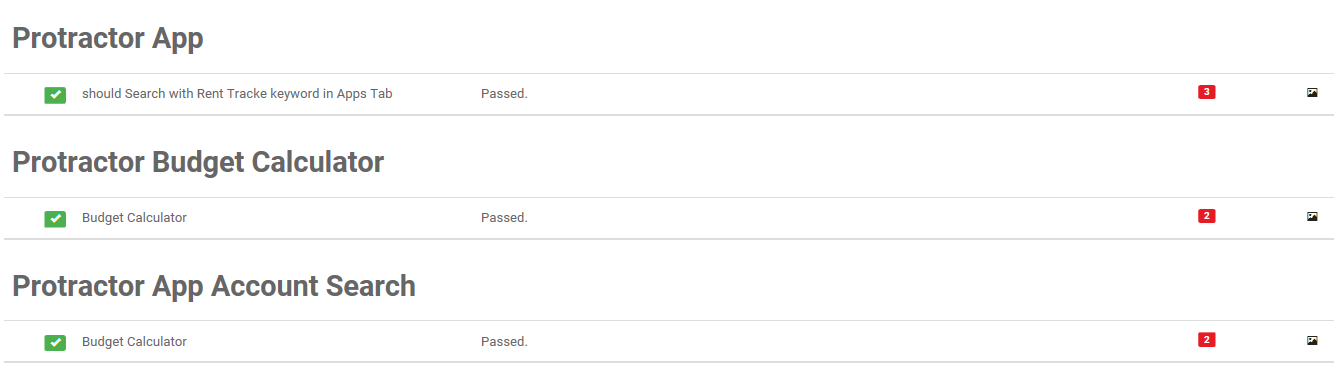


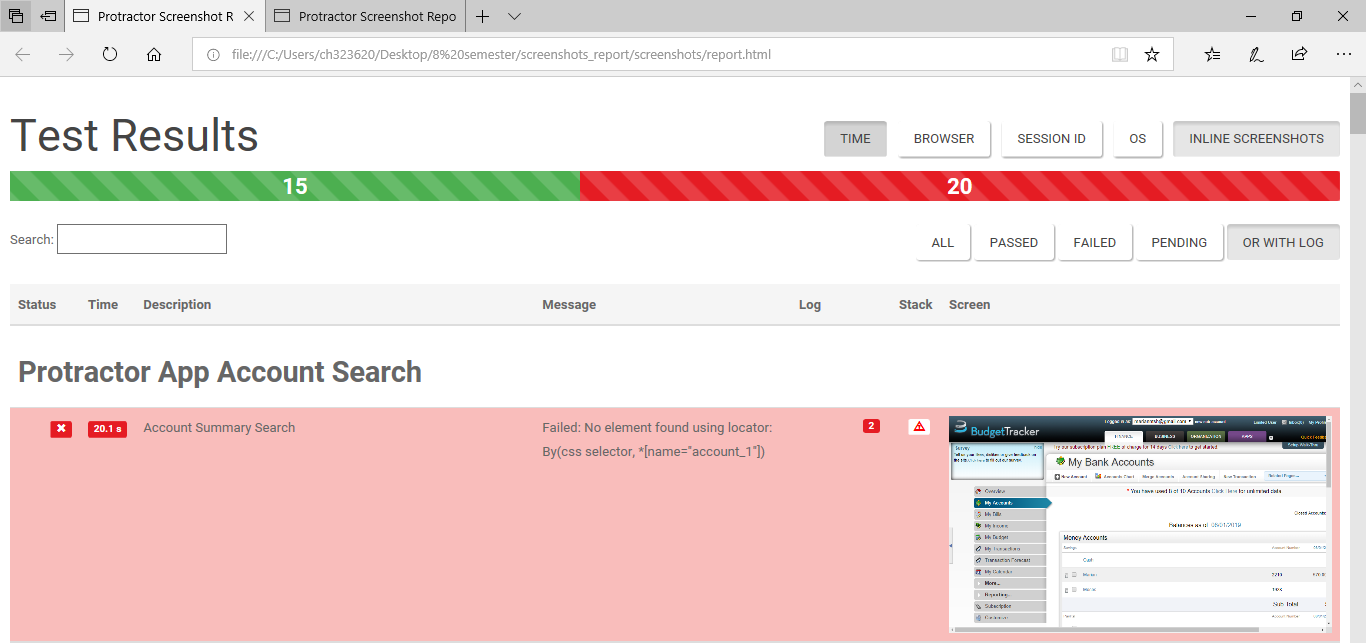








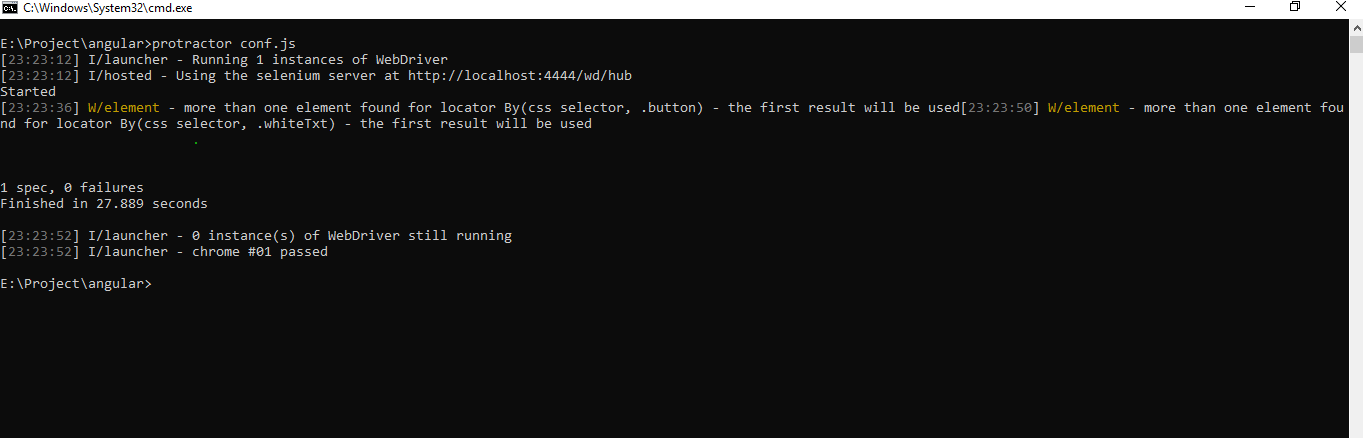


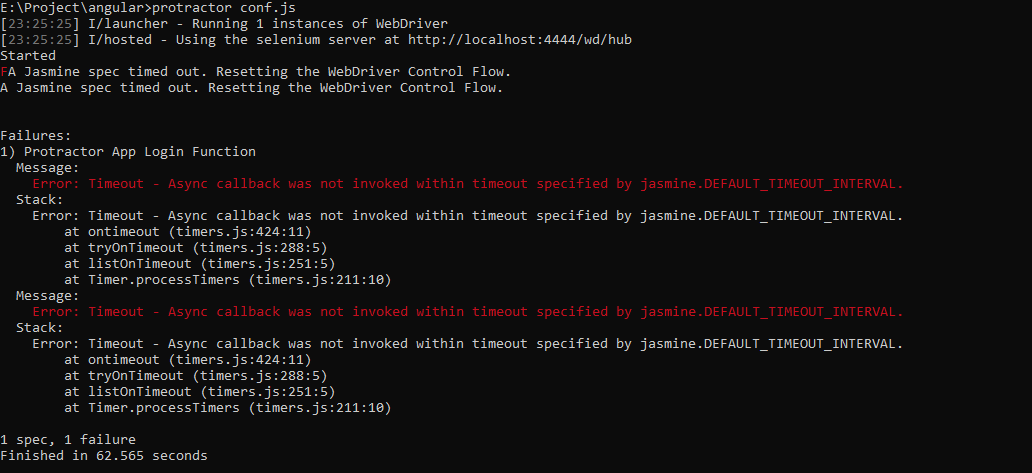


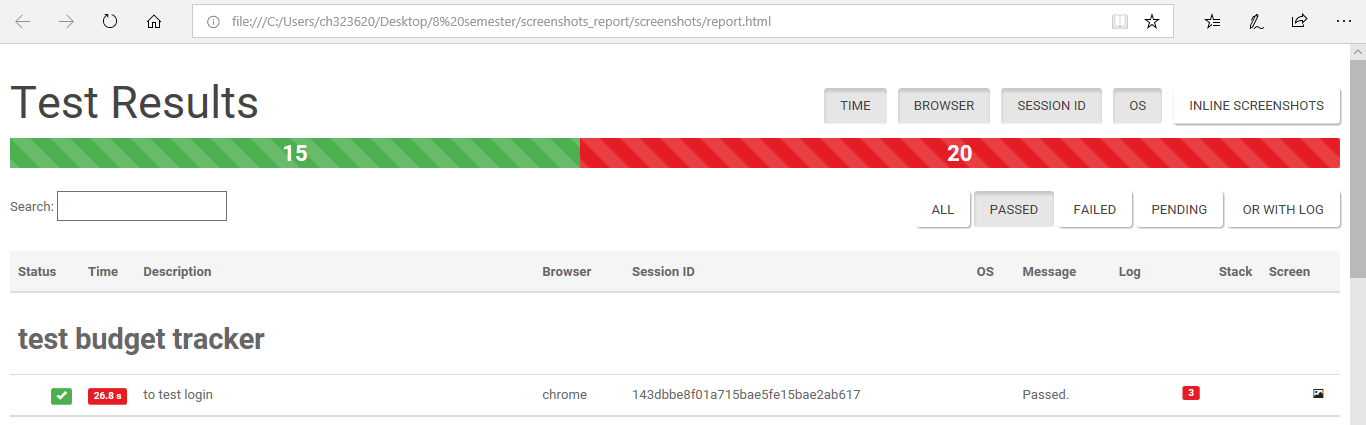
# **RESULTS AND DISCUSSIONS**

1. **Login User Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Value** | **Expectation** | **Results** |
| **Login** | The email entered should be valid | Pass | Homepage displayed for the user  User cannot  Login |
| The Password should be correct | Pass |
| The Email ID or the Password is incorrect/does not exist | Failure |

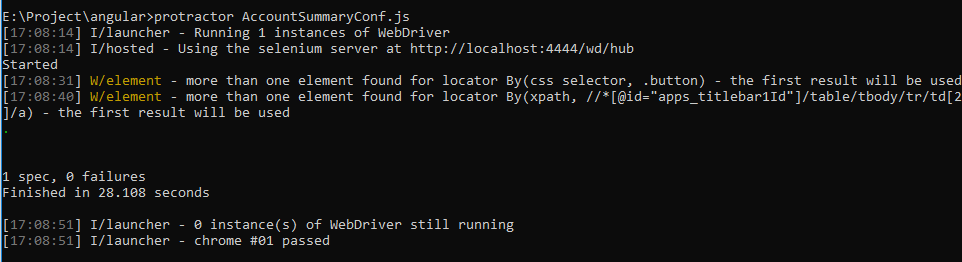


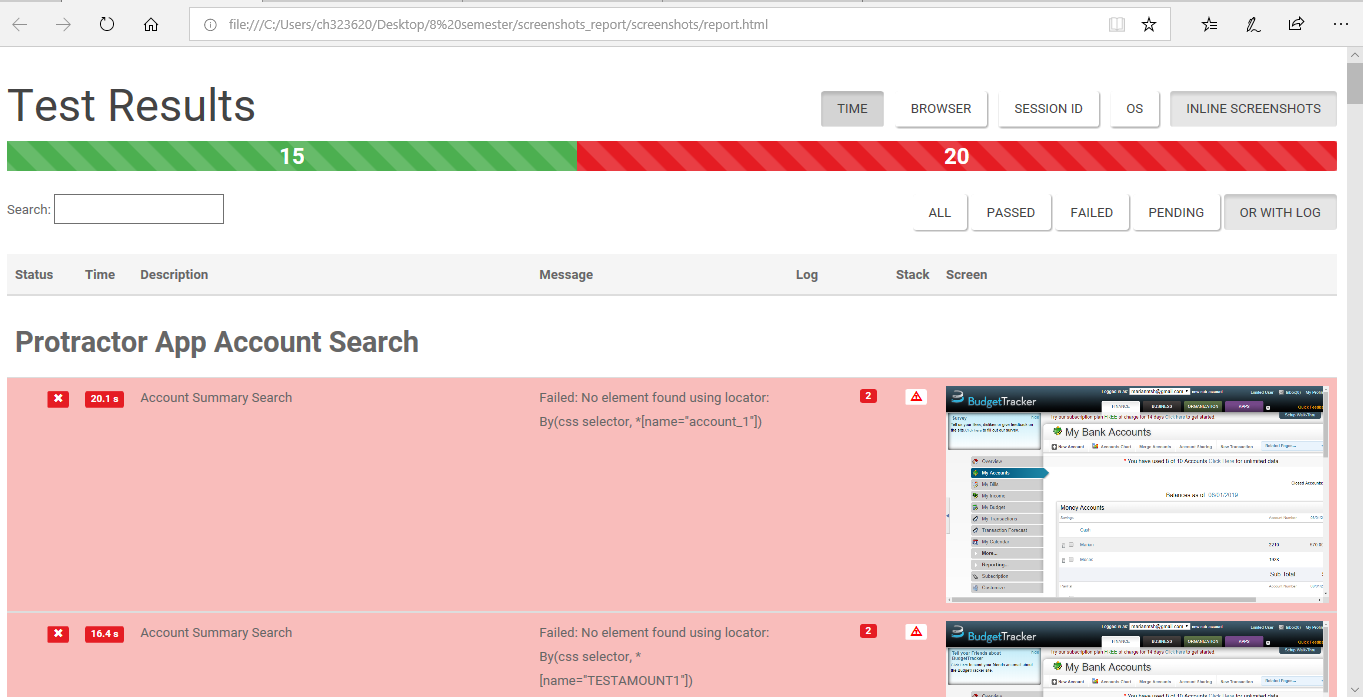




1. **Account Summary Test Case**

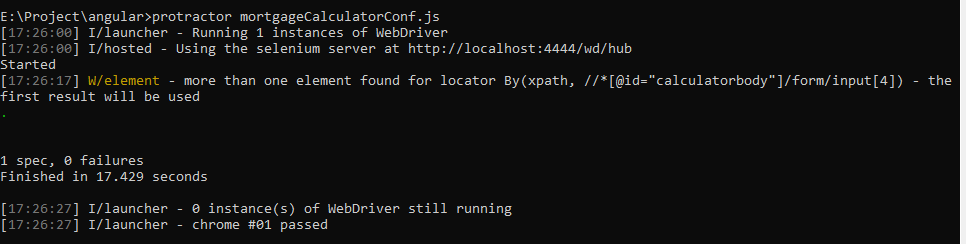
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Value** | **Expectation** | **Result** |
| **Account Summary** | It should be the sum total of all the account added under the user | Pass | The total passed as a value matches the total from the homepage |
| The sum total should not match the total from the test case | Failure | The total passed as a value matches the total from the homepage |

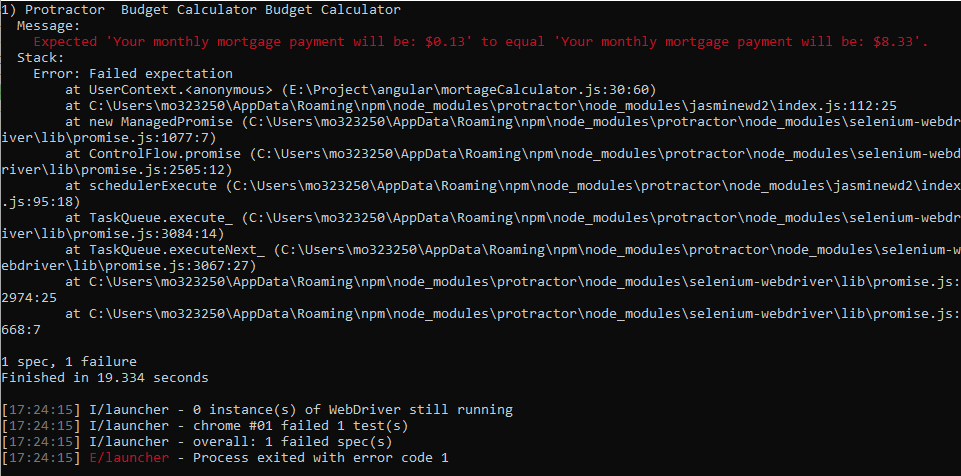


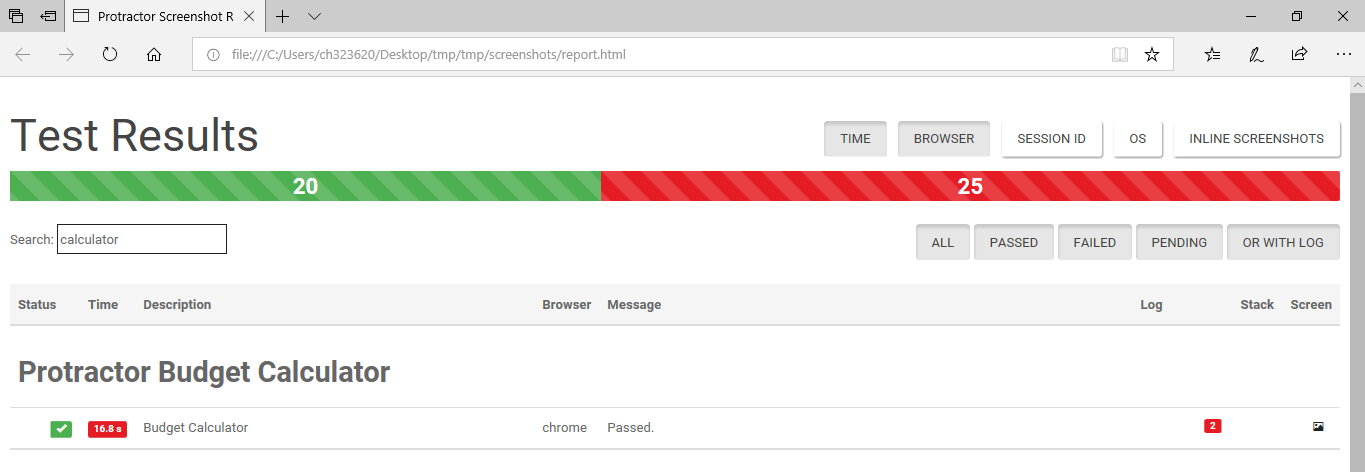


1. **Mortgage calculator Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Value** | **Expectation** | **Result** |
| **Mortgage**  **Calculator** | The amount should tally with the amount run in the test case. | Pass | The total passed as a value matches the total from the homepage |
| The total should vary from the actual amount | Failure | The total passed as a value matches the total from the homepage |

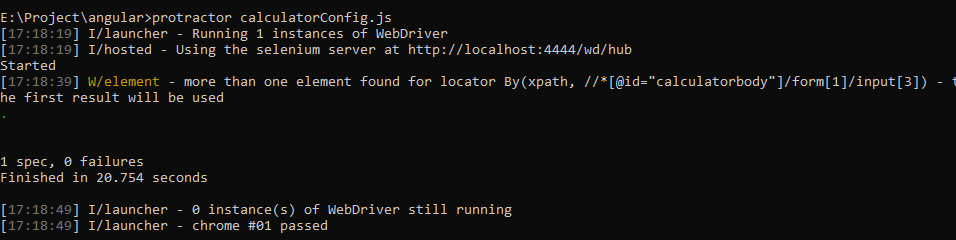


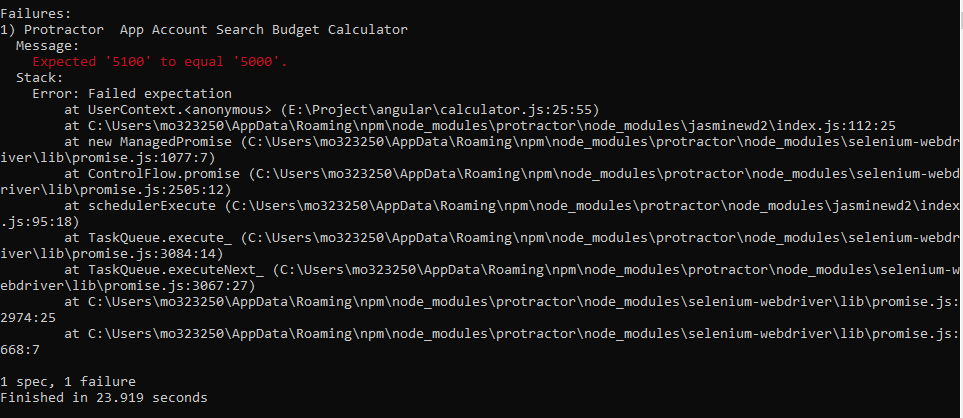


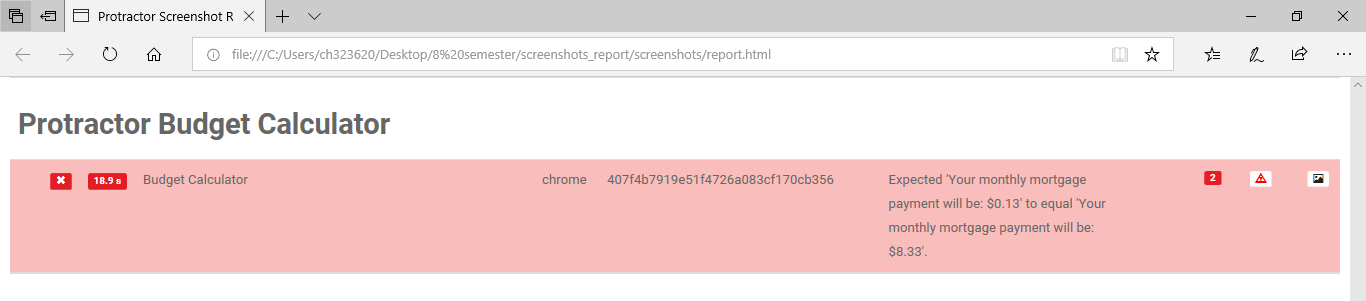


1. **Calculator Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Value | Expectation | Result |
| Calculator | The total passed as a part of the addition of the total sum of amounts should | Pass | Pass |
| If there is no value present, it should fail | Failure | Failure |





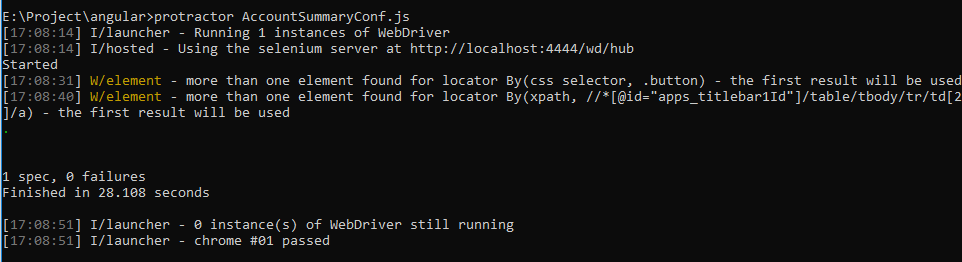


1. **Delete the record Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Value** | **Expectation** | **Result** |
| **Delete the record- check box button** | The record to be deleted should be present in the table. The check box should be selected for deleting the record | Pass | The record should be deleted |
| The record is not present in the table | Failure | The test case should throw a failure |

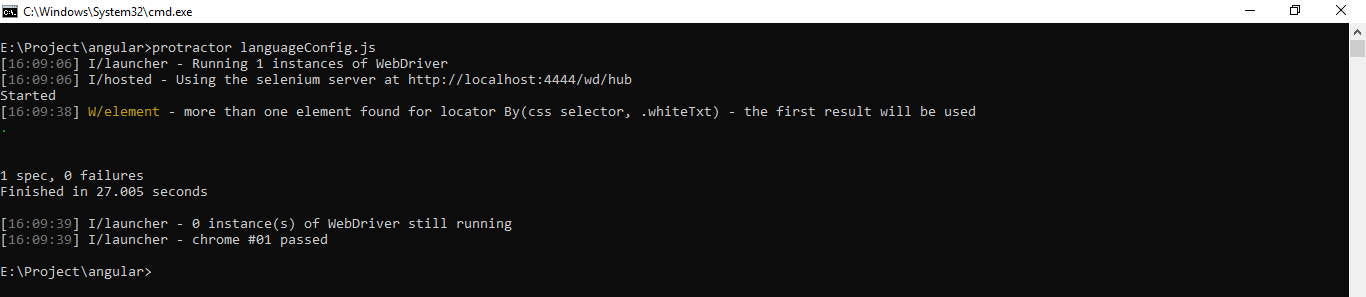
1. **Account Summary Test Case**

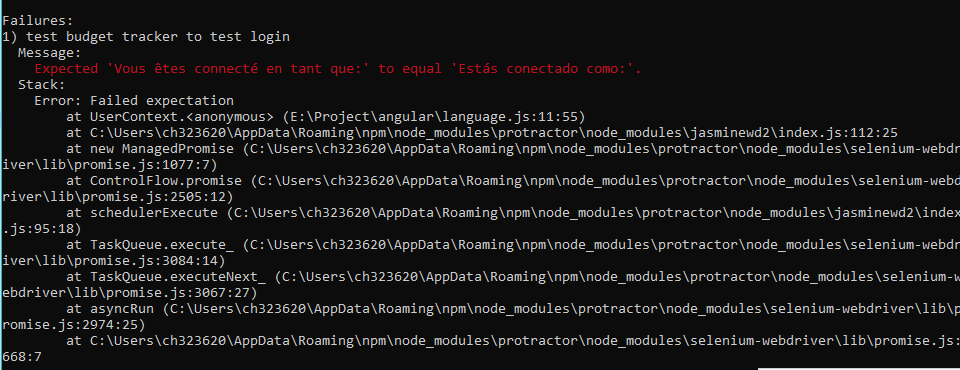
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Value | Expectation | Result |
| Account Summary | The account summary page should display the sum total of all the accounts present in the list | Pass | Pass |
| The account summary value Should fail since the value passed in the test case does not match each other | Failure | Failure |



1. **Language Test Case**

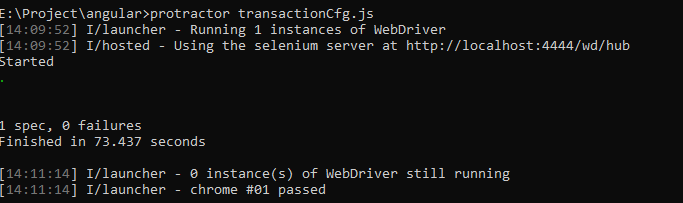
|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Value | Expectation | Result |
| Language | The language, when selected as Spanish should run in Spanish language | Pass | Pass |
| If the language passed is French, it should throw an error | Failure | Failure |





1. **New Transaction Test Case**

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Value | Expectation | Result |
| Adding a new transaction | While adding the transaction, if the account is not present it should throw an error | Pass | Pass |
| If the account Is present, the transaction should be successful | Pass | Pass |



# **CONCLUSION AND FUTURE WORK**

One of the major advantages of learning the testing framework and automation was achieved while working on this project. Apart from understanding the technicalities of the code, the loopholes in the application was also found and will be reported in the future to avoid the basic errors in an application that can be widely used otherwise.

We can conclude that Protractor is not a separate tool, but an extra layer on top of the Selenium that can handle the code/program/application that Selenium would not be able to handle with a limited support to the applications.

1. **REFERENCES**
2. <https://www.w3schools.com/nodejs/>
3. <https://www.protractortest.org/#/tutorial>
4. <https://www.draw.io/>
5. <https://www.protractortest.org/#/infrastructure>
6. <https://www.w3schools.com/js/js_json_intro.asp>
7. <http://www.tothenew.com/blog/why-use-protractor-for-end-to-end-testing-of-angularjs-applications/>
8. <https://www.guru99.com/protractor-testing.html>
9. <https://www.protractortest.org/#/plugins>
10. https://www.npmjs.com/package/protractor-angular-screenshot-reporter