CLEARSCORE QA TECH TEST

Solution

Hema Nalabanda

(hemanalabanda@gmail.com)

# Introduction

Purpose

The purpose of this document is to describe the steps necessary to install the software and make it operational. The document also provides the solutions for the three tests mentioned in ClearScore QA Tech Test.

# Installation

Pre-Requisite

1. Install Node.js

*Node will be installed in C:\Program Files\nodejs*

*Next create a new System Variable in Environmental Variables called NODE\_HOME with the value C:\Program Files\nodejs*

1. Install editor Visual Studio Code

Extract the zip file to a folder of your choice in your machine

# Folder Structure

The Project folder

The zip file has a folder called CypressAutomation that has the folder structure as shown below.

|  |  |
| --- | --- |
|  | **The folder cypress/fixtures**  *has to two json files*   * *contains test data used by Test3*   **The folder cypress/integration**  *has two .js files*   * *Task1.js – all test cases for Test 1* * *Task2.js- all test cases for Test 3*   **The folder cypress/support**  *has commands.js file*   * *defines custom methods used by Task3.js* |
| - |  |

Custom Commands

I have created two custom methods in the folder cypress/support/command.js

* verifyInvalidEmailID *(command accepts an email id and types it into the emailID field, verifies that an appropriate error message is displayed and the Get Started button is disabled)*
* verifyValidEmailID *(command accepts an email id and types it into the emailID field, verifies that no error message is displayed and the ‘Get Started’ button is enabled)*

These commands are called by the tests defined in Task3.js

# Tasks – Test 1

The Task

Using your tooling of choice, visit the ClearScore website & write tests to check that:

* the “We use cookies” notification is present
* the “We use cookies” notification can be dismissed
* the “We use cookies” notification does not reappear after being dismissed
* the appropriate cookie(s) are set

Solution

The solution has been provided in the file ClearScore\CypressAutomation\cypress\integration\**Task1.js**

How to run

Extract the zip file

Open Visual Studio Code

Select menu option ‘Open folder’ and select the folder **CypressAutomation** that was unzipped

Open a new Terminal from menu option Terminal>New terminal

To run the test, you can either

*Type the command* ***node\_modules\.bin\cypress open –*** *which will open the TestRunner. Click on Task1.js file to run the test*

*or*

*Type in the command* ***./node\_modules/.bin/cypress run --headed --browser chrome --spec "cypress/integration/Task1.js"*** *– which will open the chrome and run the test.*

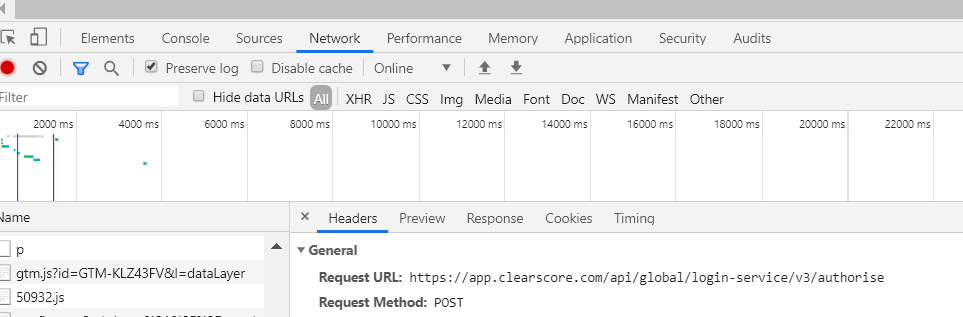
The Test Cases

1. Verify that "We use cookies" notification is present the first time
2. Verify that  "We use cookies" notification can be dismissed
3. Verify that dismissing the notification sets appropriate cookies
4. Verify "We use cookies" notification does not reappear

# Tasks – Test 2

I have used the Postman tool to test the login API

The API URL and the Request payload was picked up by attempting to login and inspecting the Network tab of Chrome Developer tools





The folder ‘PostmanAPI’ in the zip file has a collection called **‘ClearScoreCollection.postman\_collection’**

Import this into Postman

The Test will send an invalid user id and password and verify that a response code of 400 with the error message ‘access\_denied’ is received back from Postman.

# Tasks – Test 3

The Task

Using your FE framework from Test 1, provide tests to address the following acceptance criteria:

* an email address is required to sign up
* a valid email address must be provided to sign up
* when a valid email address is provided, and Sign up is clicked, the user is taken to step 1 of registration

If there are any additional scenarios you’d like to provide please do so.

Solution

The solution has been provided in the file ClearScore\CypressAutomation\cypress\integration\**Task3.js**

How to run

Extract the zip file

Open Visual Studio Code

Select menu option ‘Open folder’ and select the folder **CypressAutomation** that was unzipped

Open a new Terminal from menu option Terminal>New terminal

To run the test, you can either

*Type the command* ***node\_modules\.bin\cypress open –*** *which will open the TestRunner. Click on Task3.js file to run the test*

*or*

*Type in the command* ***./node\_modules/.bin/cypress run --headed --browser chrome --spec "cypress/integration/Task3.js"*** *– which will open the chrome and run the test.*

The Test Cases

1. Verify "Get started" button is disabled when the Sign up page is loaded
2. Verify "Get started" button remains disabled when country is selected and the email id field is not filled
3. Verify an email id is required for Sign up

*Tested by entering a blank string*

1. Verify "Get Started" button remains disabled for all invalid email addresses

*Testing with test data provided in the file* ***InvalidEmails.json***

*Testing with following data*

1. "plaintext"
2. "@gmail.com"
3. “[#@%^%#$@#$@#.com](mailto:#.com)”
4. "email@domain@domain.com"
5. Verify that an enabled "Get Started" button gets disabled when valid email id is changed to an invalid address
6. Verify "Get Started" button gets enabled for valid email addresses

*Testing with test data provided in the file* ***ValidEmails.json***

*Testing with following data*

1. " email@domain.com"
2. " firstname.lastname@domain.com"
3. “email123@domain.com”
4. " [firstname-lastname@domain.com](mailto:firstname-lastname@domain.com)"
5. Verify clicking on the "Get started" button takes you to Step 1 of the registration along with the correct country code that was selected during the sign up

# Additional Questions

Alongside the code you’ve provided for the previous questions, please provide written answers to the following questions:

1. How would you integrate the tests you have written into a continuous integration environment?

**Description of how to integrate the tests with Jenkins CI Tool**

Open the package.json file and set the scripts as highlighted

{

  "name": "cstest",

  "version": "1.0.0",

  "description": "cs test",

  "main": "index.js",

  "scripts": {

    "test": "node\_modules\\.bin\\cypress run"

  },

  "author": "",

  "license": "ISC",

  "devDependencies": {

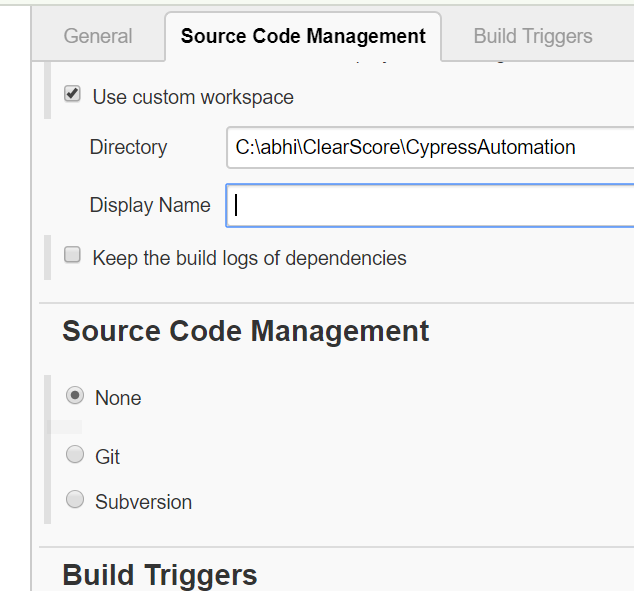
    "cypress": "^3.7.0"

  }

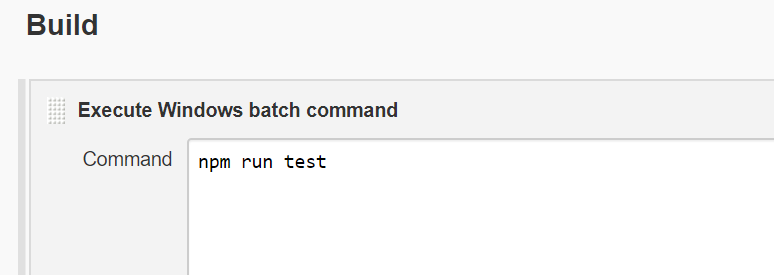
}

Open Jenkins and create a new job.

Select the folder containing your cypress tests as shown below or if the code is in Git, then select Source Code Management/Git and enter the Git Repository URL



Next in the Build section select Windowsa batch command and enter the command as shown below where test is the name given in package,json file



Now Jenkins has been configured to run the tests.

1. How would you approach running the same tests across mobile devices and different browsers?

Currently Cypress only supports Chrome, Canary and Electron. There will be support for IE, Firefox etc in the future

Currently to run the test using chrome I would give the command as shown below , where you specify the browser with - - browser chrome

***./node\_modules/.bin/cypress run --headed --browser chrome --spec "cypress/integration/Task1.js"***

If I were to implement this test in Selenium, which supports browsers like Firefox and IE, I would

create a Properties file and have the browser declared in it

In the code instantiate the selenium webdriver depending on the value set in the properties file

Public void Login(){

If(prop.getProperty(“browser”).equals(“firefox”)

{

driver=new FirefoxDriver();

}

else If(prop.getProperty(“browser”).equals(“chrome”)

{

driver=new ChromeDriver();

}

else

{

driver=new InternetExplorerDriver();

}

}

1. If you had more time available, are there any improvements that you would make to the code provided?

* I would test invalid and valid email addresses with exhaustive test data. Currently, I have only tested with four test data each.
* I would implement Page Object Patterns

*In Page Object Patterns , you create a class for every page*

*For eg., create Homepage.js inside cypress/Integration folder and create a class called HomePage in it as shown below.*

class HomePage

{

getCookieDiv()

{

return cy.get("div[class^='cookieNotice']")

}

}

In the Task1.js instead of the code below

cy.get("div[class^='cookieNotice']").should('have.length',1)

*You need to do the following*

const HomePage=new HomePage //creating an object of the class

HomePage. getCookieDiv.should('have.length',1)

* I would implement Environment variables. Nothing should be hard coded in the code. Currently my code uses hard coded URLs.

*An environment variables are declared in cypress.json file*

{

”env”:

{

“url”: "https://www.clearscore.com/"

}

}

And in the test instead of

cy.visit("https://www.clearscore.com/")

I would instead say

cy.visit(Cypress.env(‘url))