**Our Mission**

Verification of functionality for the given scenarios on ecommerce application using cypress automation tool. To achieve this I have selected page object model technique and BDD framework using cucumber both altogether.

**Project Details**

|  |  |
| --- | --- |
| **Project Name:**Ecommerce System | |
| **IDE** | Visual Studio |
| **Browser** | Chrome, electron , mozilla, edge(installed on system can be used) |
| **Tool** | Cypress |
| **Platform** | Windows |

**Cypress Framework**

Cypress framework is a JavaScript-based end-to-end testing framework built on top of Mocha – a feature-rich JavaScript test framework running on and in the browser, making asynchronous testing simple and convenient. It also uses a BDD/TDD assertion library and a browser to pair with any JavaScript testing framework.

Two primary features developed specifically for Cypress framework are:

* **Automatic waiting** – Cypress waits for the elements to become visible, the animation to complete,DOM to load, the XHR and AJAX calls to be finished, etc. Hence, there is no need for implicit and explicit waits to be defined.
* **Real-Time Reloads** – Cypress is intelligent enough to know that after saving a test file (xyz\_spec.js file), the tester is going to run it again. So, it automatically triggers the run next to the browser as soon as the tester saves their file. Hence, there is no need to manually trigger the run.

Features of Cypress Framework

* It waits to execute the test command automatically and enacts assertions before executing the next commands to be run.
* It captures screenshots during test case execution to help with debugging.
* It debugs test execution more efficiently since it provides the interactive test execution with Test Runner and logs.
* Automatically reloads Test Runner once the changes have been made to the test.

**Advantages of Our Framework**

* It can be integrated with any CI tool **Jenkins** equipped for headless execution with command-line options with multiple parameters
* It captures screenshots of test execution automatically in case of test failure. This is helpful to diagnose bugs and debug.
* Automatically reloads Test Runner once the changes have been made to the test.

**Installation of Cypress Testing Framework**

Cypress can be installed using Npm package manager as shown below:

npm install cypress --save-dev

This command will download and configure Cypress on the system.

**Components of Framework**

The following components play a part while executing particular test cases:

* **Spec File**: Contains the It(){} blocks where test execution begins. Every spec file has one describe {} block and in which different it{} blocks are contained.
* **Page Object File**: Lists the methods that contain tests’ business logic. These methods comprise the actual Cypress commands to interact with the real application.
* **Page Selector File**: Contains the actual locators of UI elements in a real application.
* **Constants File**: Used to hold any constants used throughout test files.

**Folder Structure of our framework**

While Cypress allows you to configure where your tests, fixtures, and support files are located. In our project we are using below structure.

/cypress

/fixtures

- example.json(Database)

- cucumber.json(to get cucumber report)

/integration

/examples

BDD(cucumber specific)

- FeatureFileStepDefinationFolder

- hooks<beforeEach/before/after>.js

- stepDefination.js

- FeatureFile.feature

- Testcase1WithOutBDD.js

- Testcase2WithOutBDD.js

/plugins

- index.js

screenshots

/app\_spec.js

- Navigates to main menu (failures).png

/support

- pageObjects(Page class folder)

- Page1.js

- Page2.js

- Page2.js

- commands.js(Reusable function)

- index.js

/videos

- app\_spec.js.mp4

/mochawesome-report

>assets

--mochawesome.json

--mochawesome.html

/reports

--cucumberTest-htmlreport.html

/cucumberTest-html-report.js

/cypress.json

/package-lock.jsons

/package.json

**Object Repository:** We are using Page Object Model. We have created pages class in support> pageObjects folder and defined objects in that class

**Test Data:** Need to mention TestData example.json in fixtures folder.

**TestScript:** Testscript are written in Cypress>integration>example folder. And Cucumber scripts are written in BDD folder. To write BDD scripts plugin need to be installed in visual studio IE

**Plugin:** Installed mochaweseome, cypress-cucumber-preprocessor, multiple-cucumber-html-reporter for reporting

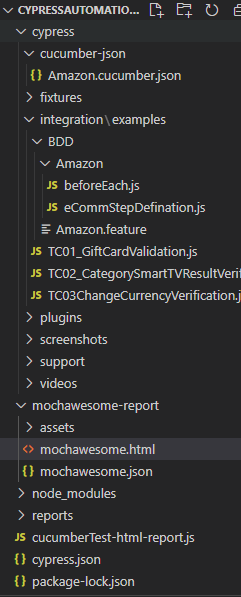
**Reports:** We are creating two type of html reports one using mochawesome and one usingmultiple-cucumber-html-reporter

**Reusable Functions:** To create function we are using support>command.js file

**Folder Structure of our project**

**Folder view**

We have created one automation suit to automate some of features of Amazon application for refrence screenshot of project heirarchy is as follows:



**Test Data:** We setup testdata

Testuser details are as below

 "username": "Shilpi Agrawal",

  "email": "automationtestingcy@gmail.com",

  "password": "Test@123",

**Mochawesome Report Generation:** After passing below command for specific testcase only one mochawesome report is generated and is overwrite the existing one in mochaweseom-report folder.

cypress run --reporter mochawesome

Some reporters accept options that customize their behavior. These can be specified in your configuration file (cypress.json by default) or via [command line](https://docs.cypress.io/guides/guides/command-line) options.

**Cucumber Report Generation:** After passing below command for specific testcase only cucumber-json file will generate

After gi

Please refer below link for details

<https://github.com/TheBrainFamily/cypress-cucumber-preprocessor>

**Limitation and scope of improvement in Project:** Reports can be more dynamic and customize format. And cucumber report need to be generate manually once run is completed could be handled auyomatically using hooks. We can add these in further improvement in project