Documentation: Automated Testing for SauceDemo

Introduction:

The task in this technical exercise is to automate the Sauce Demo page using the Cypress Cucumber tool (Gherkin language).

Cypress Cucumber Preprocessor is a plugin for the Cypress testing framework that allows you to write end-to-end tests using the Cucumber testing framework's Gherkin syntax.

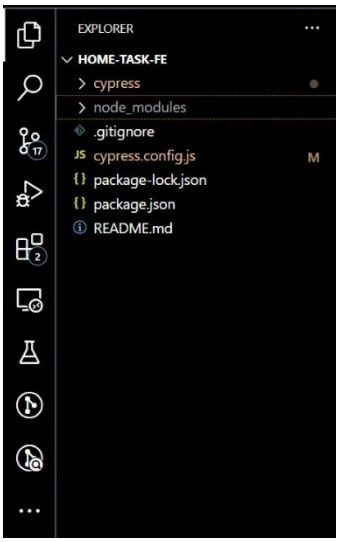
To this end, you will be asked to implement 3 test cases, 1 of which is optional.

Executions des Instructions:

I set about understanding the architecture as a whole: the connection between the Gherkin (Cucumber) language with cypress and the Page Object Model structure in the project.

So the project comes with a cypress Folders, a node_module Folder, gitignore files, cypress.config.js file, package-lock.json file, package.json file and a README.md file.

I started by opening the README file to learn about the project and install npm i, then i ran the different commands npm run test and npm run test:open.



Package-lock.json, *Package.json* files connect cypress and cucumber (Gherkin).

cypress.config.js defines configuration within the cypress project.

the base URL is set by **baseUrl**: 'https://www.saucedemo.com/'.

line 16 - **excludeSpecPattern**: '*.js

Exclude all files with *.js endings from the **integration folder** for Test scenario execution (COMMON folder).

line 17 - **specPattern**: 'cypress/integration/**/* .{feature, features}'

This leaves only the files in the **integration/AUTHENTICATE** folder, which will be the only ones to communicate the Test scenarios.

Unfortunately, I was not able to identity the configuration linking the features files with the corresponding javaScript files.

Structure POM is implemented here so that the Fixtures → helpers file contains the basic functionalities. page initiation and wait time.

the Fixtures → pages folder contains the page skeleton and the definition of queries.

the Fixtures → elements file contains the various elements that the Sauce Labs page i'll need for the test scenarios.

Fixtures → authorization folder here I use the javaScript language to define the methods that will be used in the various test scenarios.

The Integration directory describes the test scenarios in concrete terms.

each scenario is defined step by step.

In the first folder integration → Authentication the scenarios are described in files ending in .feature or .features. Here, the stages of a scenario are described in short, precise sentences. In some cases, these short sentences contain key words.

integration → common, here each file corresponds to a test.

in this file, we invoke the method(s) that make up each test step.

Approach:

my approach consisted in defining in test case 1, test case 2 and test case 3

- first, to define the various stages of my test using short phrases in the Authentication .feature files

- then translate each of these sentences into methods in my Testcases in the common files.

- then, in the authorization folder, I declare the method as it will be used in the common files.

all that remains is to furnish this or these method(s) with the appropriate elements or userData for correct execution.

I had difficulties to read errors in the cypress environment console and understanding some actions in element folder.

Example: `const [getPageName, setPageName] = useState("Home");`