Software QA Engineer

Test Automation Exercise (CGM)

Project Name: Sauce Demo Test

Prepared By: Ali Fethi Bozkurt

Project Purpose: Testing E2E flow of SauceDemo shopping experience.

Project Tools: Java, Selenium, Cucumber BDD (Gherkin), IntelliJ

Project Outputs: HTML Reports, Screenshots

GitHub Link: https://github.com/alifethi/SauceDemo.git

Introduction:

The project was done by using Cucumber BDD testing tool which uses Gherkin language to simplify to understand the testing steps. As a building algorithm, I injected first the ready classes/methods to the framework and then created Runner, and config classes. After creating the basic framework structure, I wrote the steps in Gherkin language in the feature files. To convert Gherkin language to the Java code, I created step_definition (steps) classes and began to form the automation part. For every page I created Page Object Model (POM) classes to make the code reusable, manageable and maintainable.

Framework / Outline:

Test

- **Java**
  - **Pages**[1]
    - LoginPage/MainPage/InventoryItemPage/Cart/CheckoutStepOne/CheckoutStepTwo
  - **Runners**[2]
    - CukesRunner/FailedTestRunner
  - **Steps**[3]
    - Hooks/LoginSteps/CheckoutSteps
  - **Testbase**[4]
  - **Utilities**[5]
- **Resources**
  - **Configs**[6]
    - Configuration.properties
  - **Drivers**

Features[7]

Login.feature

Checkout.feature

Description:

The folders with the red color are ready methods, coming from my existing library

[1] Pages folder contains the classes that are designed as POM (Page Object Model) to provide code usability

[2] Runners folder includes the runner classes to execute the test by correlating the Feature Folder and Steps Folder

[3] Steps folder includes the classes those generated from executing the feature files. Hooks class consists of before and after methods for each test step

[4] Testbase folder provides driver and initialize the pages in pages folder

[5] Utilities folder consists of common methods like sendtext, click, takescreenshot and configsreader class which reads and gets the values from configuration.properties file

[6] Configs folder includes Configuration.properties file that provide us to use the centralized information (url, username, password, etc) in key-value structure

[7] Features folder contains feature files written in Gherkin language.

Project Steps:

1. Login Steps:

a. Valid username and password

User enters valid username and password and I validate that user is logged in

b. Valid username and empty password

User enters valid username but empty password and validate that an error message is displayed

c. Invalid username and password

User enters invalid username and password and I validate that an error message is displayed

2. Shopping and checkout steps

a. After logging in (defined as a common step -Background- for each scenario), user clicks on an item and a new window is opened and user clicks on "Add to Cart" button.

b. User goes to shopping cart page by clicking on the "shopping cart" button and clicks on "checkout" button.

c. A new checkout window is opened and user enters the name and zip code information and clicks on "continue" button and I verify the item name on the page matches the expected item name.

d. If it matches, user clicks on the "finish" button. A new window is opened and it is verified that the success message matches the expected success message.

e. Shopping process is checked and verified end-to-end

Summary:

To test the Sauce Demo website, first I needed to test it manually and clarify the steps. After finding the structure stable enough, I automated the test steps. Since the website is not complicated, it could be quicker to automate it with basic steps, but building a more comprehensive framework by using Page Object Model, Utilities, Test Base, Configuration Options Classes, etc can provide a better and comprehensive functionality for the detailed, continuously growing requirements. But as it is obvious that there may be many other ways/options to build such framework. The usability, manageability, maintainability are the main focus for this framework. The test scenarios were selected to show the E2E testing. The framework also allows the user to derive many other scenarios