Planit Technical Assessment – Automation

Overview

You recently applied for a role of Automation Test Engineer and have been shortlisted.
The following exercises will help us determine you level of expertise in modern web automation practices.

Instructions

Using an open source UI test automation tool of your choice (Selenium/Cypress/Webdriver.io) with your choice of language and framework, automate the following test cases.

We are looking for your ability to structure your code correctly, identify the elements of a page, design the tests for scalability, and prepare the solution to be executed within a continuous pipeline (i.e. Jenkins or other)

DO NOT implement a BDD/Gherkin solution, as we will like to assess your experience with framework implementation below the BDD layer.

Application URL: http://jupiter.cloud.planittesting.com

Test cases

Test case 1:

1. From the home page go to contact page
2. Click submit button
3. Validate errors
4. Populate mandatory fields
5. Validate errors are gone

Test case 2:

1. From the home page go to contact page
2. Populate mandatory fields
3. Click submit button
4. Validate successful submission message

Note: Run this test 5 times to ensure 100% pass rate

Test case 3:

1. From the home page go to shop page
2. Click buy button 2 times on “Funny Cow”
3. Click buy button 1 time on “Fluffy Bunny”
4. Click the cart menu
5. Verify the items are in the cart

Test case 4: Advanced

1. Buy 2 **Stuffed Frog,** 5 **Fluffy Bunny**, 3 **Valentine Bear**
2. Go to the cart page
3. Verify the price for each product
4. Verify that each product's sub total = product price * quantity
5. Verify that total = sum(sub totals)

Implementation Considerations

- You will need to create a project in with minimal dependencies
- The project should **NOT** use Cucumber (gherkin - BDD)
- The tests should target a browser of your choice (e.g. Chrome, Firefox)
- The tests should run using Command Line Interface to support CI/CD

- **In the next sprint, a new version is created with new toys, different ordering of toys, and new features.**
  Please take this into consideration when designing your solution
  - **https://jupiter2.cloud.planittesting.com/#/shop**

Sharing Instructions

Please complete the solutions and share via a public repository in GitHub.

Questions

1. What other possible scenario's would you suggest for testing the Jupiter Toys application?

2. Jupiter Toys is expected to grow and expand its offering into books, tech, and modern art. We are expecting the of tests will grow to a very large number.
   1. What approaches could you used to reduce overall execution time?
   2. How will your framework cater for this?

3. Describe when to use a BDD approach to automation and when NOT to use BDD