Hello Ryan,

First of all Thanks for this wonderful opportunity. In this process with limited time there is so much to learn and improve on many things which we wont be aware of. I have run all the test cases and they are working fine.

Please see my assumptions and information on the test cases I have included in the automation. I have written this is Ruby-Cucumber framework. I have another document attached named “Setup” which gives you all the details about setup. It’s a simple setup and can be done in a few minutes.

1. There are so many test cases which I can automate other than what I have already automated. Due to the time constraints and generalizing the tests I have made an attempt to touch most of the areas although majority of the test cases are on “Enroll Now” functionality.
2. Also, I wanted to put all the selectors in a single page and access it from that page by initializing and loading the Yml page. But due to the time constrains I could not do that and I had to write all the selectors in the code itself. Its just that it would have taken a bit more time since I had to write selectors in a different page. But this can be done easily so that the code looks neat.

Please see the explanation for each test case. I have commented in the code as well but just to have a clear understanding briefing out the test cases here

Test Case 1.

* With the help of APIs I have verified all the hyperlinks present in the webpage.
* By using a gem called ‘HTTParty’, I have got the response and I match all the hyperlinks in the response and open all of them one by one.
* Since the webpage was full of links, verifying links though APIs is faster and efficient

Test Case 2.

* This test case verifies the end to end flow of Enroll now and purchasing with one of the plans.
* This test case ALWAYS FAILS because I have not provided any valid data here since there is no other way to provide genuine data and the credit card would be charged! 😊 So this fails as a “Run Time Error”.
* This is a happy path test case and It is supposed to pass with valid credentials.

Test Case 3.

* This test case verifies that all the mandatory field must be filled before purchasing a plan.
* Here I have missed couple of fields and it evaluates the invalid\_fields and passes if it stays on the same webpage but fails the test if it navigates to the “Successfully Submitted” page.
* Again the assumption here is the url changes once its navigates to the “Successfully Submitted” page which I may not be able to validate at this time

Test Case 4.

* This test compares and validates the price plan rates from one page to another
* Here I have evaluated for only one of the price plans. We can evaluate the same by using different selectors for different plans.

Test Case 5.

* This test case validates important contents on the webpage. I have selected “Installation” and “Contact” details.
* Installation is very important and we can evaluate by downloading the content on a client machine. But that needs a whole new framework like device side automation. So I have just evaluated the content on the UI.
* Contacts are very important for a company! So I have validated whether the contact number and email etc are as expected.

Test Case 6 & 7

* This test case validates both valid and invalid credit card details generated by a random credit card generator gem.
* The card generator generated both valid and invalid numbers. Again, valid number always fails since we cannot give the correct CVV or other details like name etc. Assuming that we give valid creds, we can pass this test. But as of now it fails.
* The ‘Luhnacy’ gem generates invalid numbers as well. And this helps in evaluating the validating whether invalid random numbers work on the page.

Test Case 8,9 and 10

* These test cases validate the Login page. Login page is very important, and I have just added the basic test cases here
* Test case 8 validates whether a user can login with proper credentials. This is expected to pass with proper creds.
* Test case 9 verifies whether the user can login with invalid credentials and whether the error message is displayed on the webpage
* Test case 10 verifies whether the password mismatch on “Enroll Now” page is handled properly and gives an appropriate message.
* Additionally, we can add to evaluate the Tooltips/Help/info messages on the page as well.

In Addition to the above test cases, we can add many more tests but since I wanted to touch the generic aspect and Enroll now is equally important, I stuck to the above tests. Other important tests are

1. Perf testing
2. Stress testing
3. Load testing
4. Browser compatibility
5. Accessibility testing

Hope this document helps in understanding the test cases. let me know if you have questions.

Thanks and Regards,

Ini