Implementation

**Test 1**:

Firstly, I had to manually navigate to Gruyere home page by launching the browser (Firefox). Then I click on the Sign Up link, enter valid username and valid password on username and password fields respectively. This was quickly followed by clicking the Create Account button. This is a regular procedure to boost my confidence with the application. Hence, I constructed the manual test steps as shown below:

Once this was achieved, I decided to build a framework from the scratch using TestNG framework.

However, the begging question here is - why use TestNG framework?

* It will allow me to create a central point (package) where I will keep all my test script
* It will allow me to be able to use reusable functions i.e. Launch Browser or Close Browser.
* I can create a global variable, that can be call on other part of the scripts
* this will make my test script to be robust/neat/simple
* I can paramatize my scripts
* Run batch scripts

Hence, I had to start by creating a BaseTest Package that holds my BaseTest. Inside the BaseTest package, I created a new class call the BaseTest, this holds all my reusable functions such as Launch Browser, Close Browser and I had to declare a global declaration for my driver. Once this was achieved, I had to create another package call the Guyere\_Test (This where all my test cases will reside). From the Guyere\_Test package, I created a new class call Gruyere and I had to use selenium function call extends to create a link between the Gruyere class and the BaseTest class. This no doubt will allow me to use some of the functions that has already been created in the BaseTest.

In the Guyere class, I created a method call GruyereSignUp which was meant to verify user sign up in to Guyere website. This was quickly followed by the @Test, a testNG feature that will allow the test to be executed (This ideally replaces the main that is associated with normal selenium script). Furthermore, I had to use web element locator to locate the SignUp link, username field, password field and create account button. Besides the element locator, I also have to create a verification point. The verification point is the actual test (failure to add verification point, means your test cases will always pass irrespective of whether you use valid input or invalid input). This is where you bring your knowledge of Java to the fore (if/else statement). If you look carefully at the script, you will notice that I had to wrap the “if/else statement” with a try/catch block. This was necessary because when I ran the test with a web object link (SignIn) it throw a NoSuchElement exception on execution. To overcome this, I have to wrap the if/else statement with the try/catch block.

This script will have to call the Close Browser function that was created in the BaseTest to close this script.

**Test 2**

Subsequently, another test was created to verify the creation of a new snippet. Instead of writing a separate script, I have to create a class call the NewSnippet and ensure that I connect it to the BaseTest by using the extends concept. Within the class, I created a method call CreateNewSnippet and within it, I use the element locator to locate the following element i.e. New Snippet link, Snippet test field and submit button. This was quickly followed by capturing the element that I used for the verification point. This time though, I used the Url address for the verification and finally, the script will have to call on the CloseBrowser method within the BaseTest to close the script.

Still on automation, I was ask to automate one other script that I think is relevant. I decided to automate the LogIn Link. I started by creating a class call GruyereLogInPage and I also extends the BaseTest to create a connection so as to enable this class to have access to reusable functions that are in the BaseTest class. I also created a method in the GruyereLogInPage class call GruyereLogIn and subsequent element locators i.e. “Sign In” link, username element, password element, submit button etc. This was quickly followed by the verification point, this time, I used the Url and the script will call the BaseTest class to close.

Now, the begging question is - why do I have to automate this test and the other test? The answer is: it reduces my effort, it is quicker, error level is minimal etc. Besides these, it is much easier to run batch test (regression test) even if the number of test cases increases.

Finally, I had to perform a manual test on the Gruyere Login link which I have already automate. The reason for this manual test is to check the time difference between automating and manual testing on the same object on the web page.

I started by identifying all the element in the logIn screen. Once this was captured, I had to do a metrics of all possible valid and invalid element that was identified. Though, this was much easier as there were only two elements and the maximum combination (test cases) will be four. This was quickly followed by the design of the test template that will hold all other element that will aid the test i.e. Test Case ID, Pre-Condition, Test Case, Test Steps, Expected Result, Actual Result, Test Data etc. Though, this was straight forward, as all the test performed passed and there was no bog.