About Framework:

**Using Selenium WebDriver (with Java) and Cucumber (BDD) Framework**

**Automation Project Architecture:**

**Project Directory:**

**logs** – Consists of log file created by using Log4j

**screenshots** – Consists of screenshots taken at the time of failure of scenario and same is used in reporting as well. Saved by scenario name.png. We can implement screenshot at success as well.

**src\main\java**

**pageobjects** – Locators of all the web elements are placed here according on which page they are appearing. Used PageFactory for this.

**resources**

**Base** – Base.java class used for initializing the driver for browser. Contains common utility functions like getScreenshot and explicitWait. In future can add more as well. (like excel reader, etc.)

**driver** – All drivers exe for Chrome, Firefox and IE.

**data.properties** – file containing the common values that are used in whole project

**XML config** – 2 xml config files for configuring reporting using extent reports and logs using log4j

**src\test\java**

**cucumberoptions** – consists of **TestRunner.java** class file. This is responsible for launching the test cases. We can launch then by using tags or without tags.

**features** – consists of Feature files which include all the Scenarios in gherkin language. It’s like simple English and anyone can understand the scenario that is being tested.

**stepsdefinitions** – consists of actual steps that needs to be executed for the test cases. It consists of all the actions like click, sendKeys, Verifying by Assertions whether the output is correct or not.

**Extent Report path** - \target\cucumber-reports\report.html

**pom.xml** – Consisting of all the dependencies and plugins used in the framework.

**testng.xml** – xml file for launching TestRunner.java file using TestNG, so that the execution can be started through Maven command which later can be used in Jenkins for execution.

**Scenario Explanation** – Brief explanation of both the Scenarios which I am testing and how I am verifying the results. (see the document)

**Features included in this Framework:**

1. **Logging is done using Log4J**. Proper logs are created after execution of Test Cases that can be used to debug if failure happens. Can change log levels as well. Currently new log file will create after 0.25 MB with date and time at the end of file name.
2. **Screenshots** – Screenshots are taken automatically after Scenario is failed and same is used in the Extent Report so that it will be easy to resolve or have a look at the time of failure.
3. **PageFactory** – PageFactory is used for Page Objects (Web Elements) of the page.
4. **AutoIT** - is used for uploading file from the Windows machine.
5. **No Hard URLs** are used and taking every value from data.properties file. So if any change comes in future we have to change it at one place.
6. Just by changing the browser value in data.properties the **same test cases can be executed in chrome, firefox, etc**. Can be executed in Chrome headless option as well.
7. **Extent Report** – Good presentable report is created at the end to show the figures. Feature/Scenarios that are run and status of them after execution. If failures are there, errors will be displayed in the report with screenshot at what point the scenario failed.
8. **Feature files can be run in parallel** – for faster execution multiple features files can be run in parallel to save time. Can be achieved by creating multiple TestRunners and pass the same in testng.xml file.
9. **Scenarios can be executed by tags** – Tags can be maintained at the time of implementation. So that if any change in execution happens appropriate test cases can be run. For example, we are running whole test suite which includes all the test cases of the website (we are using tag - @Web), but if we want to run only the regression suite for the website (we should change the tag to @Regression – we should already have the regression tag). Just one change in TestRunner file will be sufficient at that time.
10. **Same Steps Definitions are used** – If in some Scenario same step have to be executed as executed in some other Test Cases. Then we can use the same steps and we don’t have to write the implementation of these steps again. For ex. In this POC – User navigating to URL and Sign In steps are used again.

**Note:** For better understanding and to show all the features of the Extent Report, I am adding 1 new Feature file in which there will be 2 Scenarios that I will fail deliberately. So that we can see if failure occurs in Scenario how will it look in the report.

1st Test Case will fail because of Assertion error. Expected result is not equal to Actual result.

2nd Test Case will fail because of No Such Element found exception. Locator is not correct for the element.