**Veeva Technical Assessment - Web Test Automation**

**Introduction**

* Veeva is a product-based company that caters to the Life Science domain.
* Veeva provides a cloud-based content management system and a suite of applications designed specifically for the pharmaceutical and biotechnology industries.
* It helps pharma companies manage sales, control operations, and follow health industry regulations.

**Task to Accomplish**

In this assessment, you are requested to develop a Test Automation framework and automate 4 test cases ( minimum 2 Test Cases) for 3 Products. And the URLs of the products are

**Core Product**: [**https://www.nba.com/warriors**](https://www.nba.com/warriors) **[will be referred as CP]**

**Derived Product 1**: [**https://www.nba.com/sixers/**](https://www.nba.com/sixers/) **[will be referred as DP1]**

**Derived Product 2**: [**https://www.nba.com/bulls/**](https://www.nba.com/bulls/) **[will be referred as DP2]**

**Instructions**

* We are looking for your ability to design and develop a Test Automation Framework from Scratch, Structure your code correctly, Identify web page elements by using right locators.
* It is also important to show what code you have developed can be executed and well structured report can be presented
* Fake Test data can be used/created for automating test cases
* Share your Solution as a Zip file OR as GitHub/GitLab/Bitbucket repository
* Share the Test Automation Artifacts of the assessment before Sunday Midnight

**Framework Design and Development**

* Develop Test Automation Framework which is Optimal and Robust
* Use Technology Stacks (but not limited to): Selenium, Core Java, Cucumber with TestNG, Maven/Gradle, Logger Library(Any),Reporting Library (Any), Rest API Library (Any)
* Design an Architecture diagram of your framework
* **Design and Execution Approach : Must to Have - Mandatory Section**
  + Maven/Gradle project with root folders as src/main and src/test
  + Feature Files, Step Definitions, Business Logics, Page Objects, Test Data and Reusable Utility Methods are independent but interlinked.
  + src/main root folder should not hold any test classes/methods and src/test should not hold any reusable component classes/methods
  + Project configuration settings and Test data should be Parameterized not hard coded
  + Usage of Static waits should be limited
  + Page locators should be organized in proper Page Object Model design pattern
  + Cucumber Runner class should be created with appropriate Cucumber Options
  + Test cases should be run parallely from testng.xml file in one or multiple browsers (Chrome, Firefox etc..)
  + Proper Report should be generated
* **Design & Execution Approach :Good to Have - Bonus Section Additional (~ 1 - 2 hours)**
  + As the assessment is to automate 3 products, design the framework in the Multi Module Maven project with 4 modules.
    - Module 1: automation-framework
    - Module 2: core-product-tests
    - Module 3: derived-product1-tests
    - Module 4: derived-product2-tests
  + The automation-framework module will hold all the reusable code and it will be shared to all 3 test modules.Test modules will hold their product specific codes that cannot be shared across.
  + Dynamic use of browser binaries(Chrome,Firefox etc..) for web driver creation
  + Zero Usage of By class, if you are designing Page Object Model design pattern
  + If you are using the Page Factory design pattern,PageFactory.initElements() should be called only in one place throughout the Framework. It should not be called in each Page class constructor.
  + Use JSON/YAML file to maintain test data
  + Use Cucumber Hooks effectively
  + Don’t pass cucumber Tags from Runner Class, Pass cucumber tags from testng.xml file and bind the tag to Runner Class during runtime
  + Generate Runner classes for each Feature file during runtime.
  + Run the test scenarios in multiple threads without using testng.xml file and by using maven commands

**Test Case(s) To Automate: Minimum 2 Test Cases**

* **Test Case 1: for CP**
  + From the **CP** home page , go to >> Shop Menu >> Men’s
  + Find all Jackets ( from all paginated pages)
  + Store each Jacket Price, Title and Top Seller message to a text file
  + Attach the text file to the report
* **Test Case 2: for CP**
  + From the **CP** home page , hover on menu Item >> click on New & Features
  + Count total number of Videos Feeds and count the videos feeds those are present in the page >= 3d
* **Test Case 3: for DP1**
  + Go to **DP1** home page
  + Below Tickets Menu, count number of slides present
  + Get the title of each Slide and validate with expected test data
  + Count how much duration each slide is playing and validate with the expected duration
* **Test Case 4: for DP2**
  + Go to **DP2** home page
  + Scroll down to the footer
  + Different links for various categories (Team, Tickets, Shop, etc..) will be visible
  + Find all the hyperlinks of the Footer links into a CSV file and report if any duplicate hyperlinks are present.

**Checklist for Assessment Submission:**

* Framework Architecture Diagram
* Automation Assessment Artifacts
* Readme file with well documented instructions
* All the project files that are required to run the tests independently in (Windows/MacOS)