**Selenium Automation Framework with Real-time Dashboard (Dockerized ELK)**

1. What are the components/tools used in the project?
2. Java, Selenium, TestNG, Maven, AssertJ, POM (Singleton and Factory design pattern), PageFactory,

ThreadLocal(for thread safety), Extent Reports 5.0.5, Excel sheets for test data, Property or json as config files, TestNG listeners like annotation transformer, ITestListener, ISuiteListener, Running test in cloud or dockerized Selenium Grid, Parallel cross-browser testing.

1. Some unique features in the project?
2. i. Real-time dashboarding with ELK images.

ii. Using sonarlint to write clean code.

iii. Selenoid Integration.

1. Issues with parallel execution in testing? What is the Thread Local class?

* When we perform parallel execution, there are multiple threads that create objects of a static class, like a driver object of WebDriver class by multiple test methods. But the class can only point to a single object at a time, and during parallel execution, multiple threads try to run commands on their own objects, which is not possible. All the execution will happen on the object of the fastest thread.
* Thread Local class: To solve this problem of threads fighting to run commands parallelly on their objects, we use the “Thread Local” class.
* Thread Local class helps to create variables that can be read and written by the same thread.
* Easiest way to safeguard static class variables.
* We create a ThreadLocal class variable of type “WebDriver” class, and use the getDriver() and setDriver(<driver>) methods to get and set the driver.

1. Property Files. Why do we need it?

* Property files are basically files containing key value pairs of the utility data we use like “browser=chrome” ,”url=’www.google.com/”etc. We need this file to avoid hardcoding the values we want to use in our test cases. We can use this file, to keep all this data and then use another class like ‘readProperty” file to pull in the data.

1. What is AssertJ? What are it’s uses?
2. AssertJ is a library we can use to write readable, fluent assertions.

* Add AsserJ maven dependency and import the library:

import org.assertj.core.api.Assertions;

* Using it on a Strings:

Assertions.assertThat(<input string to be checked>)

.isNotNull()

.isNotEmpty()

.isEqualTo(<string>)

.contains(<string>)

.containsIgnoringCase(<string>)

.containsWhitespaces(<string>)

.doesNotContain(<string>)

.as(<string>).isNotEmpty()

.endsWith(<string>)

.hasSize(<number>)

.hasSizeLessThan(<number>)

.hasSizeBetween(<num1>,<num2>)

.matches(<regex\_pattern>)

.as(“<error message>”).isEqualTo()

.as(“<error message>”)..containsIgnoringCase (); etc.

We can have such multiple fluent assertions in a single statement.

* Using it on integers:

Assertions.assertThat(<input no. to be checked>)

.isEqualTo(<number>)

.isBetween(<num1>,<num2>)

.isLessThanOrEqual(<number>)

.isGreaterThanOrEqual(<number>)

.isEven()

.isPositive()

.isInstanceOf(Integer.class)

.isCloseTo(<num>, Percentage.withPercentage(<percentage>)

.isCloseTo(<num>, Offset.offset(<offset>))

.as(“<error message>”).isEqualTo()

.as(“<error message>”).isEven (); etc.

* Using it on lists:

Assertions.assertThat(<input list to be checked>)

.contains(<element>)

.extracting(<element> -> <element>.getText())

.doesNotContain(<element>)

.isNotEmpty()

.hasSize(<num>)

.hasSizeGreaterThan(<num>)

.hasSizeBetween(<num1>,<num2>)

.startsWith(<element>)

.containsExactlyInAnyOrder(<element1>,<element2>….)

.doesNotContainAnyElementOf(<list>)

.doesNotHaveDuplicates()

.as(“<error message>”).contains()

.as(“<error message>”).isNotEmpty(); etc.

* Using it on hashmap:

Assertions.assertThat(<hash map to be checked>)

.doesNotContainEntry(<key>,<value>)

.containsEntry(<key>,<value>)

.containsKey(<key>)

.doesNotContainKey(<key>)

.containsValue(<value>)

.hasSize(<number>)

.isNotEmpty(); etc.

* We can also use AssertJ assertions with Custom class objects, data objects etc.

1. What is Page Object Design Pattern? What are the advantages and disadvantages.
2. Advantages: Effective Segregation of Page Objects from tests, Functional Encapsulation, Scalable, Reusable etc.

Disadvantages: High setup time, Complexity and Ambiguity in implementation.

1. What is Dataprovider in TestNG? How to use it?
2. DataProvider is a feature of the TestNG library that allows a developer to run the same suite of test cases with different data sets. We can use it for data driven testing.

Eg: @Test(dataProvider=”getData”)

public void test(<data provider method return type> <var>){

…..

}

@DataProvider

public <return type> getData(){

…….

}

The data provider method can return a 1-D/2-D array of any type. We use Apache POI libraries to capture the excel sheet data and return it to the test method taking the data in.

Data Provider with json:

For getting data from json file, we use the Jackson JSON library. We have to install two dependencies: “jackson-core”, ”jackson-databind”. In the data provider method, we create hashmap to store the json k-v pairs and return a 1-D object containing these hashmaps.

1. Listeners in TestNG. What are they? What’s the use?

Ans. ITestListener, ISuiteListener, IMethodInterceptor, IAnnotationTransformer, IRetryListener,.