

High level Description and Control Flow

Date: 17/01/23

**DOCUMENT CONTROL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Prepared by** | Amarendra Pani | **Designation** |  |
| **Date** | 17/01/2023 | **Signature** | N/A |

|  |  |  |  |
| --- | --- | --- | --- |
| **Reviewed by** |  | **Designation** |  |
| **Date** | MM/DD/YYYY | **Signature** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Approved by** |  | **Designation** |  |
| **Date** | MM/DD/YYYY | **Signature** |  |

**DOCUMENT CHANGE HISTORY**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Version** | **Section** | **Description** | **Author** | **Reviewer** | **Approver** |
| 17/01/2023 | V0.1 |  | Initial Draft | Amarendra Pani |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Automation Framework Description and Control flow.**

1 Framework layers and its use

2 Control Flow

3 Example

# 

# Framework layers and its use

1. In Page layer we are maintaining all the Elements with By locator strategy and Created all the Public Page Actions and methods by using selenium commands.

2. Factory layer will produce the driver and initialize property (init\_prop) method from Driver factory will read configuration properties from each environment File.

3. In Object repo layer we store all the configuration level data’s as key and value pair in properties File.

4. In Test layer we created Separate Test Page Classes and TestNg Assertation must be sure for each and every individual test methods.

5. In Base Test layer, setup() and teardown methods are defined, which helps the driver to initialize the browser and every test class must be extend with Basetest.java class

6.In Utilities and libraries layer we created the Actions method by selenium, JavaScript executor and created Excel utils for supporting the data driven testcases.

7. The Test data layer have the Excel sheet workbook and sheets for Storing and supporting Data driven Test.

8. Test Reports and logs layer contains all the latest Results links (Extent report, Allure report, TestNG Report) and screenshot for pass, fail and Skip.

9. In Test Runner layer we created separate Testng.xml files for Regression, Smoke, functional and Master Suite.

10.Maven is Responsible for handling the dependencies and helped for build, clear ,Deploy with fat jars.

**2.Control Flow**

1. Executing the TestNG.xml file.

2. Maven will Start the Build, Clear and Call the testing.xml suite.

3.Testng.xml file call the Test methods from Test page layers.

4. All the Test Page classes must extends the Base test class, in which setup() and tearDown() Methods available, by using @BeforeTest and @AfterTest annotations.

5. The setup() from BaseTest class call the init\_driver() method, which is available in Driverfactory class.

6. The init\_driver() call the init\_Prop() method and this method will get the environment set-up values from the properties file and start the browser.

7. Driver will focus to the Page classes in which the element locators and Public page methods available.

8. Driver call the public methods and the methods call utilities methods for the Actions class and the action class will return some positive response and negative response to the methods which are available in the Page class.

9. Each and every public method from page class will return some value.

10. Those value will move to the Test Page methods and do the validation by using the Assertation by TestNG.

11. After the execution it will generate the Results with all the screenshots.

# 3.Example

1. In our Project when we run the testing.xml file from test runners package, maven build life cycle started in backend and it will call the testNg.xml file.

2. In testing.xml file, it call the test modules as per the test page classes like (HomepageTest.java, Contactpagetest.java…..) which are available in src.test.java folder.

3. All the Test.java classes extends the BaseTest.java Class.

4. In BaseTest.java class SetUp(), and teardown() available with @BeforeTest and @AfterTest annotation by testing.

5.The setup() method call the DriverFactory.java class which available in src.main.java -factory package.

6. In Driverfactory the init\_driver() is there and it call the init\_prop().

7. The init\_prop() call the enviroments like browser, url, username, password from properties file .

8. Now the Browser Start-up as per the value from properties file.

9. Then the driver will focus the individual page class like HomePage, contactPage, Dynamic365Page etc………

10. In those page classes we created the element locators and public page methods like

Eg- Homepage.java class(getHomePageTitle(),getHomePageUrl(),isRbtLogoExist())

11. Those Methods will responsible to call the locators as per the requirement and also call the Actions methods like sendKeys(), click(). From Utils package in ElementUtil.java class.

12. And every Public Page class methods (getHomePageTitle(),getHomePageUrl(),isRbtLogoExist()) will return a value.

13. And all the Testpage classes calls the methods with the assertations by using the values from Public page class methods.

14. After the execution when we refresh the project, all the reports generates successfully with the screenshots from testOutPut folder and screenshots stores in screenshots folder.

……………………………………………………………………………………………………………………………………………….All the above’s are short descriptions.