Page No.
Date

Selenium framework -

* What is selenium framework →
  - ✓ selenium framework is a code structure that makes code maintenance easy & efficient.
  - ✓ Without frameworks, users may place the "code" & "data" at the same location which is neither reusable nor readable.
  - ✓ Framework produce beneficial outcomes like increased code reusability, higher portability, reduced cost of script maintenance, better code readeability etc.

Types of framework -

There are mainly three types of frameworks created by selenium webdriver to automate manual test cases.

1) Data driven
2) Keyword driven
3) Hybrid

1) Data driven framework

- ✓ Data driven framework in selenium is a method of seperating test data from the test case.
- ✓ Once the test data are seperated from the test case, it can be easily modified for a specific functionality without changing the code.
- ✓ It is used to fetch Test Data from external files like Excel, .csv, .xml or some database tables.

Page No
Date

* POM with pagefactory →

POM - Page object model

It is a java design pattern use for design classes in Test script.

Page object model is an object design pattern in selenium, where

1) Webpages are represented as classes &
2) The various elements on the page are defined as variables on the class.

In this we use pagefactory to initialize webelements (variables) that are defined in webpage classes or page objects (classes).

POM - Incapsulation

POM strictly follows incapsulation concept where,

1) Datamember (DM) should be declared globally with access level private.
2) Initialize within a Constructor with access level public using pagefactory.
3) Utilize within a method with access specifier public

Note -

1) No. of D.M. that need to be created under a pom class will depends on number of elements that need to be handle in a webpage.
2) pom class will not contain a main method, to run a pom class we require another class with main(), i.e. test class.
3) Test class will contain all the navigation steps to test an application.

Page No.
Date

POM class →

1) Pom class depends on webpage present in an application
2) For each webpage a seperate pom class will be created.
3) Number of pom class depends on the number of webpages in an application
4) In each pom class D.M./variables created using encapsulation ~~method~~ concept and initialized by using pagefactory
5) Number of D.M. created in pom class will depends on number of elements present in a webpage.
6) Each declared D.M. should initialized & utilized in pom class.

Test class –

1) Test class depends on number of test cases written by manual test engineer.
2) Test class will contains navigation steps & inputs that need to be given to the components/elements.
3) In test class data/inputs that can be given directly or through external sources like Excel sheet.

Pagefactory →

✓ It is a class which contains static method like initElements().
✓ To initialize D.M./variable in pagefactory we need to use initElements method within the constructor.
Syntax – PageFactory.initElements(driver, this);
✓ initElements() will initialize D.M. by identifying each component present in a webpage by using @FindBy annotation. which takes locator type as an input.
Syntax - @FindBy(locatortype = "locator value/expression")
Private WebElement DM;
eg @FindBy(xpath="") Private WebElement UN;

Page No.
Date

Working of PageFactory →

1) While executing Test script initElement method will convert all the datamembers @FindBy annotation to findElement() this process is known as basic/eardy initialization. - After ~~creation~~ creating object of pom class.

@FindBy (xpath = ' ') private WebElement PN →
private WebElement PN = driver.findElement(By.xpath(" "));

2) To perform action on component we need to call a methods

3) Before performing each action initElement method will identifies component present or not, then it will do complete initialization this process is known as late/lazy initialization.

Disadvantage of POM

✓ POM will initialize all the component before performing actions, but sometimes application may contains few components which will be hidden & displayed once we perform action on components, that hidden component will not be displayed while pom initializing so it throws "No such element" exception.

✓ To overcome drawback of pom, we need to use "PageFactory" is an extension of pom.

Page No.
Date

Difference between POM & PageFactory.

POM	PageFactory
1) It will initialize all the D.M. present in class ~~before class~~ completely before performing action on components.	✓ It will initialize the D.M. present in a ~~performing~~ class before performing each action
2) It will use if webpage is not containing hidden elements	✓ It will be used if webpage is containing hidden elements

* Advantages of POM

- ✓ Makes the code reusable
- ✓ It is helpful in reducing code duplication
- ✓ It makes ease in maintaining the code (flow in the UI is seperated from verification)
- ✓ Makes code readable (Methods get more realistic names)
- ✓ The code becomes less & optimized.

POM method name steps -

1) Action
2) Classname
3) element name

} Eg - public void setKiteLoginPageUsername()

1) Actions - Titles/names & purpose
   - i) Set - to enter values in a field
   - ii) click - to click on a link or button
   - iii) Verify - to verify testcase

Page No.
Date

TestNG

TestNG → (TDD framework - Test driven development framework)

TestNG is a java unit framework used for writing / designing of Test classes.

It is used to design / script only test class and can not be used for pom classes.

Installation process - 1) Click help
2) Click eclipse market place
3) Search TestNG in search box
4) Choose TestNG for Eclipse
5) Click install & restart Eclipse.

A class designed / scripted using TestNG do not have public static void main() method

eg- public class classname
{
@Test
public void m1()
{
}
}

1) Emailable report →

✓ Report generation is very important when performing Automation testing as well as for manual testing.
✓ Just by looking at the result, you can easily identify how many test cases are passed, failed and skipped.
✓ By looking at the report, you will come to know what the status of the project is.
✓ Selenium web driver is used for automating the web-application, but it won't generate any reports.
✓ The TestNG will generate the default report.

Page No.
Date

steps to generate Emailable report -

1) Execute test class & refresh the project.
2) You will get test-output folder.
3) In that folder right click on the "emailable-report.html" & select the option open with the web browser or double click on it.

Note -

1) If we use sop() to display text as a output then result will be in console & not in Emailable report
2) To display text in Emailable report we need to use static method log present in Reporter class
   eg - Reporter.log("String msg", true);;

2) Annotation →

i) @Test ii) @BeforeMethod
iii) @AfterMethod iv) @BeforeClass
v) @AfterClass.

i) @Test →

✓ It is used for execution of test method/test case.
✓ Method execution if a class having more than one method happens in alphabetical order if priority is not defined

ii) @BeforeMethod -

✓ Preconditions required for a test case are deffined with this annotation.
✓ It will executed ~~each &~~ before each & every test case in a class repetedly/seperately if multiple test cases are there.

Page No.
Date

iii) @AfterMethod →
- ✓ Post conditions of a test case are defined with this annotation
- ✓ It is also executed multiple times with each & every test case if multiple test cases in a class seperately

iv) @BeforeClass →
- ✓ It is used for execution of test method before execution of test class
- ✓ It is executed only once (at start) in a test class

v) @AfterClass →
- ✓ It is used for execution of test method after execution of test class
- ✓ It is executed only once (at end) in a test class

eg-

```
public class classname
{
  @BeforeClass
  public void m1(){ }
  @AfterClass
  public void m2(){ }
  @BeforeMethod
  p. v. m3(){ }
  @AfterMethod
  p. v. m4(){ }
  @Test
  p. v. m5(){ }
  @Test
  p. v. m6(){ }
  @Test
  p. v. m7(){ }
}
```

execution sequence ↓

m1()
m3()
m5()
m4()
m3()
m6()
m4()
m3()
m7()
m4()
m2()

Page No.
Date

* ~~Keywords for TestNG~~

vi) @BeforeSuite -
It will run only once, before all tests in suite are executed

vii) @AfterSuite - It will run only once, after all test in suite are executed

viii) @BeforeGroups - This method will run before 1st run of that specific group

ix) @AfterGroups - This method will run after all tests from specific group get executed

x) @BeforeTest - This will be executed after all test methods in the current class have been run.

xi) @AfterTest - when all @Test annotation methods complete execution of those classes inside the <test> tag in the TestNG.xml file.

Suite → Groups → Test → Class → Method → Testcase

Page No.
Date

3)* Keywords for TestNG –

i) invocationCount
ii) Priority
iii) enabled
iv) TimeOut
v) dependsOnMethod

i) invocationCount →

Sometimes a testcase need to execute multiple times which can be possible by using TestNG keyword invocationCount

Syntax: @Test (invocationCount = Number); → no. of times testcase execution

eg. @Test (invocationCount = 5); → 5 times execution

ii) priority →

✓ To change test method/ testcase execution order/sequence, we need to use TestNG keyword "priority".

eg - ~~priority=1~~ @Test (priority = 1)

✓ Priority can be negative, positive and duplicate value
✓ If priority is not mentioned then by default it will be treated as zero
✓ From low number ~~prio~~ priority to high number priority test method/testcase execution takes place.
✓ Incase of priority with duplicate number is mentioned then execution takes place by alphabetical order.
✓ Priority never be a) decimal value
b) variable

Syntax: @Test (Priority = number)

eg- ① @Test (priority = 5)
@Test (priority = -2)

Page No.
Date

iii) enabled →

✓ In certain condition we have to disable a test case that can be achieved by using enabled keyword.
✓ Disabling a test method / test case in TestNG can be achieved by setting the enabled attribute of the @Test annotation to false.

syntax

```
@Test (enabled = false);
```

iv) TimeOut →

✓ If a test class contains many methods / test cases & one of the test case takes long time / time consuming, in such case we need to fail that test case and execute next remaining test case.
✓ We have to set time limit for test case execution if test case is not completed in this time limit then it will be skipped and treated as failed.

syntax-

```
@Test (TimeOut = milliseconds);
```

eg- `@Test (TimeOut = 5000);`

v) dependsOnMethod →

✓ If execution of one test case is depends on execution of another test case / multiple test case then we have to use dependsOnMethod keyword of TestNG

eg-①

```
@Test()
public void loginToApp() { }
@Test (dependsOnMethod = {"loginToApp"})
public void logoutToApp()
```

②

```
@Test (dependsOnMethod = {"condition1", "condition2"})
```

Page No.
Date / /

4) Test. Suite →

- ✓ It is xml file which contains all the test classes names which need to be executed.
- ✓ It is used to execute all/multiple Test Classes.

syntax

```
<suite name = "Suitename">
<Test name = "Test name">
<classes>
<class name = "packageName1.className1"/>
<class name = "packageName2.ClassName2"/>
</classes>
```

step to get suite file →

Right click any class → TestNG → Convert to TestNG → xml file → if required change name of xml file → Finish → file → open source → run it with adding other classes in its code as per requirement.

Page No.
Date

5) Assert Class →

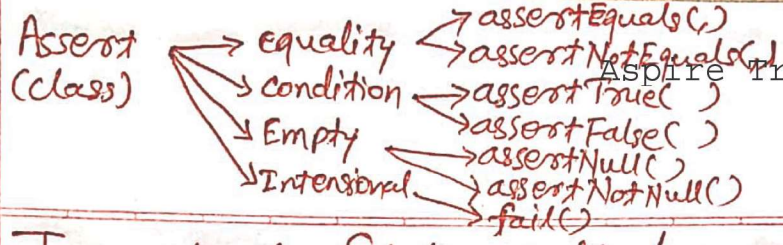
Verifications using TestNG →

Example to verify checkbox is selected or not

```
Class Sample
{
main()
{
System.setProperty(" ", " ");
WebDriver driver = new chromeDriver();
driver.get("URL");
WebElement rv = driver.findElement(By.xpath(" "));
if (rv.isSelected())
{
sop("checkbox is selected");
}
else
{
sop("checkbox is not selected");
}
```

Verification steps

- If above verification process is used to verify expected result of a test script case, length of Test script will be increase & Test script will take more time for execution.
- To reduce length of test script we need to use Assert class for verification which contains static methods.



Page No.
Date

Important Static method present in Assert class →

All static method should be imported from org.TestNG.

i) assertEquals()
ii) assertNotEquals()
iii) assertTrue()
iv) assertFalse()
v) assertNull()
vi) assertNotNull()
vii) fail()
viii)

i) assertEquals() →

- ✓ It is used to verify expected & actual results.
- ✓ If both results are same/equal then o/p is Pass otherwise Fail

eg - @Test
Public Void Test1()
String str1 = "Hello";
String str2 = "Hi";
Assert.assertEquals (str1, str2);
(actual result, expected result)
(classname)
o/p ⇒ Fail

ii) assertNotEquals() →

- ✓ It is used to verify expected & actual result.
- ✓ If both results are not same/equal then test case get Pass otherwise Fail

eg - Assert.assertNotEquals (str1, str2);
o/p ⇒ Pass

iii) assertTrue() →

- ✓ This method is used to verify conditions are true or false.
- ✓ If condition is true o/p is pass otherwise fail.

eg - Assert.assertTrue (rv.isDisplayed)
Assert.assertTrue (rv.isSelected)
Assert.assertTrue (rv.isMultiple)

Page No
Date

iv) assertFalse ()→
✓ It is used to verify conditions are true or false.
✓ If condition is true o/p will be Fail otherwise Pass.
eg- Assert.assertFalse (False)
o/p ⇒ Pass

v) assertNull ()→
✓ This method is used to verify components or text fields empty or not, ~~if it is empty o/p~~
✓ If it is empty then o/p is pass otherwise fail
eg- Assert.assertNull (rv);

vi) assertNotNull()→
✓ This method is used to verify components or text fields are empty or not
✓ If it is not empty then test case is pass & if it is empty test case fail.
eg- Assert.assertNotNull (rv);

vii) fail ()→
✓ This method is used to intensionally identify fail test method
✓ In certain condition we have to fail test case with by intension, so we use fail() method
eg- @Test
public void m1()
{
Sop ("Hi");
Assert.fail()
}
o/p ⇒ Fail

Page No.
Date

In a test class if one of the test method is fail then TestNG will stop execution of failed test methods & other test methods execution will be continue & dependent test methods will be skipped.

eg. @Test
public void Test1()
{
Reporter.log("Test1", true);
Assert.fail();
}

@Test(DependsOnMethods="Test1")
public void Test3()
{
Reporter.log("Hello", true);
}

@Test
public void Test2()
{
Reporter.log("Test2", True);
}

o/p ⇒ Test1 – Failed ⇒ Pointed in red
Test2 → Passed ⇒ Pointed in green
Test3 → skipped

* Disadvantages of Assert class →
If a test class is containing multiple test methods, in one of the test method having multiple verifications.
If one of the verification is failed then rest of the verifications will not be verified & TestNG will execute next method by failing verification field method.

eg- public class sample
{
@Test
public void m1()
{
String str="Hello";
Assert.assertEquals(str,"Hi");

Page No.
Date

```
Reporter.log ("Hello", true);
String str2 = "Hi";
Assert.assertEquals (str2, "Hi");
Reporter.log ("Hi", True);
}
@Test
public void m2()
{
Reporter.log ("TestNG", True);
}}
```

Skipped without verification

O/P ⇒

Passed	Failed
0	0

TestNG ⇒ O/P

* SoftAssert →

✓ To overcome above drawback we need to use softAssert.

✓ It is a class which contains non static methods which are used to do verification

✓ SoftAssert will do verification if any verification is failed, notifies & executes the rest of verification in a test method.

eg -
```
@Test
public void m1()
{
String str = "Hello";
SoftAssert soft = new SoftAssert();
soft.assertEquals (str, "Hi");
Reporter.log ("Hello", True);
String str2 = "Hi";
soft.assertEquals (str2, "Hi");
Reporter.log ("Hi", True);
soft.SoftAssertAll(); }
```

Page No.
Date

@Test
public void m2()
{
Reporter.log("TestNG", True);
}

o/p ⇒ Hello
Hi
TestNG

✓ We have to intentially write assertAll() method so that it will execute all methods
Soft.assertAll();
✓ If we don't write assertAll method then it will not notify.

Hardassert	Softassert
✓ If assert statement fails it will immediately throw AssertException & test suite continues with next @Test	✓ It collects error during @Test & does not throw any exception & executes next statement of same method
✓ It fails test case & remaining statements will be aborted.	✓ It starts executing next statements
✓ object not required to use it	✓ object creation is required

Page No.
Date

* Failed.xml →

While executing the automation scripts, test cases may fail for several reasons. To optimize our next runs we need to rerun only failed test cases.

Steps to execute failed.xml file -
1) Create testing.xml file under project folder
2) execute testing.xml file
3) In the test-output folder "testing-failed.xml" file will be created automatically
4) To rerun execute only 'testing-failed.xml file

Possible reasons for test case fail
1) Environment issue
2) Script error
3) bug

Page No.
Date

* Disable test case execution (skip test case without execution)

1) Two ways -

i) from test class using 'enable=false' keyword

eg -

```
@Test (enable = false)
public void TC()
{
}
```

ii) from suite using include/exclude keywords

eg -

```
<suite name="Suite">
  <test name="Test">
    <classes>
      <class name="Packagename.classname"/>
        <method>
          <include name="TC1"/>   <= TC1 will be executed
          <exclude name="TC2"/>   <= TC2 will not execute
        </method>
    </classes>
  </test><!--Test-->
</suite><!--Suite-->
```

Page No.
Date

* Grouping of test cases -
TestNG Groups allow you to perform grouping of different test methods -

eg -

```
public class class1
{
@Test (groups="orders")
 public void TC1() { }

@Test (groups="setting")
 public void TC2() { }
@Test (groups="setting")
 public void TC3() { }
```

```
public class class2
{
@Test (groups="fund")
 public void TC4() { }
@Test (groups="setting")
 public void TC5() { }
@Test (groups="setting")
 public void TC6() { }
```

Suite code for grouping

```
<Suite name="Suite">
<test name="Test">
 <groups>
  <run>
   <include name="setting">
  </run>
 </groups>
  <classes>
   <class name="TestNG Packagename.class1"/>
   <class name="TestNGpackagename.class2"/>
  </classes>
</test> <!-- test -->
</Suite> <!-- suite -->
```

@ Here, TC2, TC3, TC5 & TC6 will be executed.

Page No.
Date

* Parallel testing or parallel execution :-

✓ parallel testing or parallel execution, as the name suggests, as it is a process of running the test case parallely rather than one after the other.

✓ In parallel testing, the program's multiple parts (or modules) execute together, saving the testers a lot of time & efforts.

eg :-

```
<suite name = "suite" parallel = "tests">
  <test name = "Test 1">
    <classes>
      ...
    </classes>
  </test> <!-- test -->
  <test name = "Test 2">
    <classes>
      ...
    </classes>
  </test> <!-- test -->
</suite> <!-- suite -->
```

here, Test 1 & Test 2 start execution at same instance so that called as parallel testing

Page No.
Date

* Multibrowser testing ~~or compatibility testing~~ -
or compatibility testing

Running same test case in two or more than 2 browsers.

Browser	driver
Chrome	chromeDriver
Firefox	geckoDriver

Eg -

Eg -

```
<Suite name = "Suite">
 <test name = "Chrome test">
  <parameter name = "browserName"
             value = "Chrome" />
  <classes>
   <class name = "packName.TC1" />
  </classes>
 </test> <!-- Test -->

 <test name = "firefox test">
  <parameter name = "browserName"
             value = "firefox" />
  <classes>
   <class name = "packName.TC1" />
  </classes>
 </test> <!-- test -->
```

```
public void TC1 (String browserName)
        throws InterruptedException
{
 WebDriver driver = null;
 if(browserName.equals("firefox")
 {
  System.setProperty("webdriver.
  gecko.driver", "path");
  driver = new FirefoxDriver();
 }
else if (browserName.equals("chrome")
 {
  System.setProperty("webdriver.
  chrome.driver", "Path");
  driver = new chromeDriver();
 }
 driver.get("http://www.facebook.com");
}
```

Page No.
Date

* Compatibility testing in parallel — or multibrowser parallel testing

Page No.
Date

* Difference bet^n Junit & TestNG -

	Junit	TestNG
1) Parallel testing	does not support	Support
2) Annotation	does not support advanced annotation	Support
3) Dependency test	does not support	Support
4) Grouping Test	does not support	Support

Page No.
Date

* POM-DDF-TestNG

Categories actions & test cases with using different annotations & pom classes by using external test data file

eg- @BeforeClass
Browser opening action
Excel loading action
Navigating to home page etc actions are included
Also creating objects of all or other classes using driver
In rest of the annotations actual test case related tasks are included

* POM-DDF-TestNG-Baseclass-Utility

Baseclass - A class in which common actions are included
Utility class - ~~Feature~~ Different utility tasks are included with static method like excel loading, capturing screenshots
So that Base classes can be inherited (extends) in test class as per requirements
& static methods of utility classes can be used as per requirement

* Extent Reporter -

✓ Extent reporter is an open-source reporting library used in selenium test automation
✓ Extent reporters becomes the first choice of selenium automation testers, even through selenium comes with inbult reports using frameworks like JUnit & testNG

steps - property object creation
file input stream
load file in property object
get value by passing key value with getProperty

Page No.
Date

* Property file selenium -

steps to create property file -
creating a properties file in eclipse

1) Right click on the main project folder & select New → other → select general → file & click on next button → provide a valid file name with extension '.properties' on the new file resource window & click on finish button

2) storing data onto properties file.
Data is stored in properties file in the form of key-value pairs, with key being unique across the file
Open file in Eclipse & store some data
eg - URL = http://kite.zerodha.com/

3) Reading data from properties file -

```
Properties obj = new Properties();
FileInputStream objfile = new FileInputStream
        (System.getProperty("user.dir") + "\\credentials.
        properties");
obj.load(objfile);
String value = obj.getProperty("URL");
```

* Capture Screenshot of failed Test Case -

listner -
To get test case status, we need to use listner (interface in TestNG)
Different types of listners are there.
ITestResult result ⇒ Test case result pass/fail
we use in @AfterMethod
Methods in ITestResult - CREATED, FAILURE, SKIP, STARTED, SUCCESS, SUCCESS-PERCENTAGEFAILURE

Syntax

```
if(result.getStatus() == ITestResult.FAILURE)
    {Utility.screenCapture.screenshots(driver, TCID);}
```

New Project → Maven → maven Project → Next →
Next →
Group id - org.apache.maven.archetypes
Artifact id - maven-archetype-quickstart

Maven Project

group id : Package name
Artifact id : project name
jar file = dependencies

Maven convention over configuration
- src/main/java - all pom classes / library files
- src/test/java all test classes
- Maven dependencies folder
- pom.xml

For every module a seperate package is created for it

Structure :→
Project
- SRC/main/java
  - Library files ← Package
    - Baseclass } classes
    - Utility class
  - Module1
  - Module2
  - Module3
  - Signin Module } Package
    - Login1 Page } classes
    - Login2 Page
- SRC/test/java
  - Test module4
  - Test module5 } Package
  - LoginTest Module
    - LoginTest1 class } classes
    - LoginTest2 class

Page No.
Date

JRE System Library
Maven Dependencies
Test-NG
Browser
Screenshot
Testdata
Folder1
Folder2
Pom.xml

Folders

→ Project object module

Pom.xml →
website → mvnrepository
search for selenium (or Test NG or any other)
select selenium as per version requirement
click
copy code & paste it in pom.xml

Page object Model (POM) – Design pattern – approach ⇒ Page chaining model
+Data driven framework
+TestNG

Framework

com.qa.base package

TestBase class

WebDriver driver; new chromeDriver()
maximizeWindow(),
pageLoadTimeout()
implicitWait(), deleteAllCookies(),
get(url)

com.qa.config.

Environment variables
config.properties
url, username, pwd,
browsers, other

Inherite Inherite

com.qa.pages Package

Page layer / page library / object repository

Login Page class
1) Web objects / web Elements
2) Actions Method Features

Home page class

Registration Page class
- click on Sign in link()
- fill reg. form()
- check logo()
- footer links()

Add to cart class

Payment class

com.qa.tests Package

Test layer

Login Page Test class

Home Page Test class

Registration Page Test class

Add to Cart Test class

Payment Test class

com.qa.testdata

excel sheets

com.qa.util

Test Util class

Screenshot(),
sendmail();
commonutil(),
Generic functions()

Test-output/reports

Test Report: Pass/fail

HTML

TestNG
XML Reports
Extent Report

Tools –

Java, selenium webDriver, TestNG (writing test cases), Maven (To creating build), Apache POI API – (To read data from excel file), Extent Report (TestNG report), Log4j API, Jenkins (continuous integration to trigger the code), GIT repository (to maintain code), GRID (- parallel testing), Browsers (ff / Chrome / IE / Safari), OS (Windows), Virtual machines (VMs / Source Labs / Browser stack / Local machines)

Contact –
8766060616, 9860994463, 9028790227

Project
↳ module (15-20)
→ Test classes (10-20)
→ Test cases (5-10)
→ Pom classes (20-40)

Total modules	15-20	15
pom classes in each module	20-40	30
Total pom classes in project	400-600	450
Test cases in each module	70-100	80
Test cases in each Test class	5-10	8
Test classes in each module	10-20	10
Test classes in project	150-200	150
Total number of test script in project	1600-200	1600
per day Test script (depend-complex)	1-4	avg 2
per month Test script (20 days)	40-50	avg 40

* Framework layout

Tool - selenium
Prog lang - Java

Maven
→ src/main/java - All pom classes & library files
→ src/test/java - Test classes
→ Maven dependencies
→ Test NG
→ pom.xml → All dependencies.
→ Browser → chromeDriver, etc.
→ Screen shots → All screenshots stored in this folder
→ Test data → Excel
→ Property file → file.properties (login credentials)

Base class
Jenkins (scheduling) ← CI tool
github repository ← To store source
Utility class ← selenium common files
Extent reporter ← Report generation
log4j jar ← Log generation

Page No.
Date

* property file –
  ✓ It is used to store important login credentials of an application
  eg- Appl^n UN, pwd, url, diff. port numbers, etc

* Log generation –
  ✓ To generate the logs in selenium framework, we need to make use of log4j jar file.
  ✓ log4j is an open source logging framework (API)
  ✓ By using log4j, we can store selenium automation flow logs in file or database.
  ✓ log4j has 3 principal components –
    1) Loggers –
       It is responsible for logging information
    2) Appenders –
       Used to deliver log events to their destination or
       we can say appenders is used to write logs in a file.
    3) Layouts –
       It is responsible for formatting logging information in different styles.

* Extent reporter –
  ✓ Extent reporter is a customizable HTML report which can be integrated into selenium webDriver using TestNG framework.

Advantages – 1) Customizable HTML report with stepwise & pie chart representation.
2) Displays the time taken for testcase execution in report
3) Each testcase can be associated with a screenshot
4) Easily integrated with TestNG.
5) Multiple testcases runs can be within suite & can be tracked easily

Page No.
Date

Sprint flow ⇒

sprint - ①, ②, ③ & ④ purely manual

sprint - 5

← 15 days Execution * 15 days Development → (20 working days)

Mail from manual team to automation T.L for automation TC development

TL mail to automation team members.

∴ TL mail → Team members

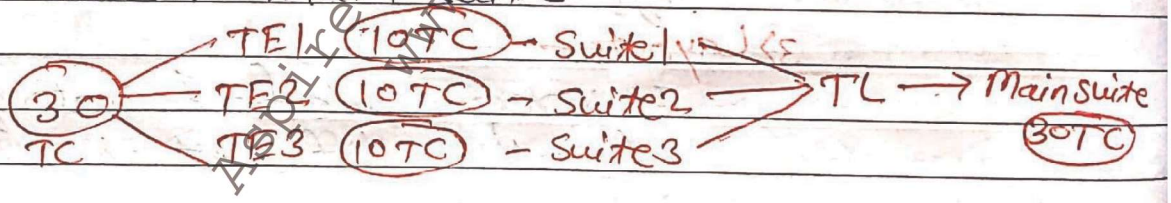
Total TC (50)

Approved (30) ← Test engineer 1 (10)
TE2 (10)
TE3 (10)

Only development of automation script (Already created framework)

1) Creating pom classes
2) Creating Test classes.
3) Preparing Test data
4) Creating common ~~data~~ files/functions
5) Revising test cases (scenarios)

After end of sprint ⑤ 30 test cases are added to main suite



Sprint - 6

← Execution 15 days * Development 15 days → (20 working days)

Mail from client → P.M. → T.E. for new sprint (end of sprint 6)

Team meeting with TL

↓

30 TC from main suite

↓

execute

Pass	22
fail	08

20 TC pass
5 fail → bug
5 TC script error → Modify → 2 TC Pass, 3 TC fail

Page No.
Date

Sr.No	T.C. Name	Priority	Status
1		1/2/3	Pass/fail/skip
2			
1			
30			

⇩

JIRA	
Task	Bug
SPL execution status	Raise bug for failed TC for sprint-5

Development - 15 days in sprint 6
mail from manual team to automation TL
70 TC → Approved 60 TC
TE1 20TC
TE2 20TC
TE3 20TC
TL ← Main suite (30 old + 60 new TC)

At end of sprint 6 total 90 TCs.

Sprint-7 execution 15 days, development 15 days
90 TC
70 Pass
10 Bug (fail)
10 script error
5 Pass
5 fail

Pass	75
fail	15

* Release 1 completed after 6 month or 1 Year (i.e. after 6 or 12 sprints)

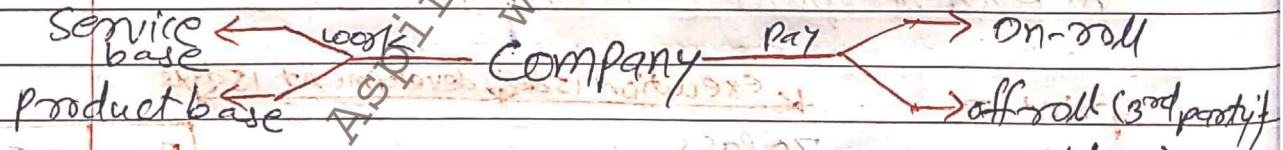
Page No.
Date

Apps - 1) Mail - Outlook
2) meeting - Skype / teams / slack
(chat / audio calling / video calling / screen share)

Leaves -			Apply ↓,
1)	Sick leave	- 7 days	- same day
2)	Casual leave	- 7 days	- 1 week before
3)	Privileged leave	- 22 days	- 2-3 week before
4)	Maternity leave	- 26 weeks (approx)	
5)	Paternity leave	- 3 to 8 weeks (approx)	

Probation - First 3 to 6 month before permanent

Awards - ~~1) Star performed~~
1) Star emp. of the month
2) Put on the back
3) Stand out performer
4) Achievers of the month



Page No.
Date

* Steps involved in automation -
1) Selection of testing tool
2) Define scope of automation.
3) Planning, design & development
4) Test execution
5) Maintainance.

* Steps involved in planning phase of automation. -
1) Selecting right automation tool
2) Selecting automation framework if any
3) List of scope for automation tool environment setup.
4) Preparing grant chart for ① development & execution.
5) Identify test deliverable.

* The condition where we can't use automation in Agile
1) When Agile testing always ask for changes in requirements
2) When exhaustive level of documentation is required in Agile.

* Primary features of good automation tool -
1) Test environment support & easy of use
2) Good debugging facility
3) Robust object identification
4) Object & image testing ⑧ ability.
5) Object identification
6) Testing of database
7) Support multiple framework

Page No.
Date / /

* Roles & responsibilities of automation test engineer

1) Selenium environment setup:
   eg- Download & install eclipse, java lang, configure selenium jar file, TestNG, Maven --- etc

2) Inspect elements / objects -
   Using Firepath or firebug (inspect in chrome)

3) Creating test case using element locators & selenium webDriver commands -
   - Element locator for identifying elements
   - Selenium webDriver for performing operations on elements

4) Enhancing test cases using program features -
   eg- flow control stat, conditional stat, exception handling, adding coments comments, error handling, verification etc

5) By using techniques -
   Grouping test cases, prioritozing test cases, executing test batches & generating test reports, using testing framework eg- TestNG. for net-nunit

6) Data driven testing, DB testing, cross browser testing
   - Executing some functionality with multiple set of data

7) Analysing test report result & reporting defects.
8) Selecting test cases for Regression testing, defect tracking
9) Regression testing on modified build.
10) final regression
11) Maintainance of test automation resources.

Page No.
Date

* List of challenges faced in project :-

1) Domain knowledge of application
2) Vast application flows, functionalities & module interconnection was complex.
3) Use of dynamic XPath
4) Handling multiple exceptions while test execution eg. StaleElementException, NullPointerException,
5) Dependencies of test cases on one another
6) Execution of single test case works (pass) but fail in suite.
7) Use of BeforeMethod implemented to minimize dependencies
8) Maintainance of test data & pre-conditions on multiple scripts
9) Multiple unexpected popup handling
10) Application freeze because of multiple users using single port
11) Many setting related test cases were tested on different port to minimize the effect on other test cases

Page No.
Date

* JDBC connection -

```
P SV main (String [] args) throws Exception
{
  // load the driver
  class.forName ("Oracle.jdbc.driver.oracleDriver");

  // get the connection object
  Connection conn = DriverManager.getConnection
        ("jdbc:oracle:thin:@localhost:1521:xe",
              "username", "password");

  // Create statement object
    Statement st = Conn.CreateStatement();

  // execute querry
    ResultSet rs = st.executeQuery ("Select * from studed");

    While (rs.next())
    {
      Sop (rs.getInt(1)+" "+rs.getInt(2));
    }
  // Close connection
    rs.close();
    St.close();
    Conn.close();
}
```

* JDBC - It is API by using this we are connecting to database
Oracle - It is database to which we are establishing the connect
thin - It is the java type 4 drivername.
@localhost - It is the database location.
1521 - It is port number of database
Xe - service name of database.

Page No.
Date

(a) Cachelookup concept -
New annotation available in pagefactory API.
@ (1) Cachelookup means it store perticular name/object in cache memory so everytime when we iteract with the same element/object get that element from cache memory so that speed of framework/script will improved. Performance of script improves.

* WebDriver FireEvent - to generate action logs.
We have to add webDrive listner class need to add in utility. it extends base class & implementing webDriverEventListner also.
It having standard template.
We have different actions in selenium, so whenever we perform a action related method will be called and concern message will be printed on console so we get proper exact log messages so to know what is happening in automation

Page No.
Date

Extent Report →

steps - 1) Add extent report dependency in pom.xml
2) Create a class of extentReportListner in src/main/java (if required in seperate package)
3) Copy extent report standard template script in this class (if required remove errors if any occured)
4) This will generate extent.html report
5) Report name can also be changed with Name.html
6) It will be generated in test-output folder
7) Before running we have to add listners for extent report in testing.xml file.
8) It gives pie charts of test case status

Add in suite listner
<listner>
<listner class-name = "Packagename.filename".
</listner>
→ extententreportNG.java

ExtentReportNG.java Template

Public class ExtentReporterNG implements IReporter {
private ExtentReports extent;
public Void generateReport (List <XmlSuite> xmlSuites, List<ISuite> suites, String OutputDirectory) {
extent = new ExtentReports (OutputDirectory + File.separator + "Extent.html", true);
}

Page No.
Date

* Jenkins *

* Install & configure jenking

Need
1) Download Jenkins war file (Jenkins.war)
2) Download from jenkins update center
3) Select latest version from list & download
4) Create a folder (at any location) named Jenkins
5) From command prompt go to folder Jenkins
6) java -jar jenkins.war presenter after typing this command so that it will install
7) In command prompt a admin user created with password, copy that password.
8) Default port for jenking is 8080 in local machine
9) Open browser type localhost:8080 in address bar
10) Unlock jenkin with admin copied password.
11) Install suggested plugin option need to select
12) Getting started page - create 1st admin user
13) Installation finished.

* Setup ~~& configure~~ a maven project -

first time setup →
1) Click on Manage jenkins
2) Manage plugins → Available → Search Maven →
3) select all maven related options → Download now & install & restart
3) Reloggin with credentials
4) Click on item name → Maven Project will be listed.
5) Enter item name (project name) → click on Maven Project → Description (if required) can be given → Maven info plugin (Config) (skip it) → Source code Mgt (None - (Local m/c, Git - git repository.) → Build trigger (keep default option (do not change anything) → Build env. environment (skip it) → Prestep (Skip it) →

H H 1,15 1-11 *
↓
hourly once a day on 1st & 15th of every month from Jan to Nov.

H H (9-16)/2 * * 1-5
↓
hourly once during 9am to 5pm from mon to fri

* * * * *
↓
every minute

Build → (Maven version - tool config. → Add maven name → given any name → Apply & save
Root pom - pom.xml (path of pom.xml file in project need to copy & paste it)
Goal & options (clean install) →
Post steps (run regardless of build result) →
~~Build step~~
Post build action → (install TestNG Result plugin to generate report) → Select option ☞ publish TestNG Results → ~~Apply~~ TestNG XML result pattern → give path of testing-results.xml file of our project (from test-output folder) → Apply & save

* Run Automation test suite from Jenking.
1) Click on project icon (named with given project name)
2) To run testing click "Build Now"
3) After finishing testing again go to project
4) Go to build history (as per requirement, mostly latest)
5) Check test result
"Report" can be emailed to manager / team lead
6) Test result trend graph is also available on screen

* Build periodically option →
steps → 1) Click project 2) Build now
3) Build Triggers 4) Build periodically
schedule → comment box syntax ⇒ each seperated by space or tab

MINUTE HOUR DOM MONTH DOW
0-59 0-23 1-31 1-12 0-7 (0 & 7 is for sunday)

~~eg - 30 21 05 08~~ Duration (n1-n2) → from to
Concern time (n1, n2) → at a perticular time, Interval (/n) → every n time eg.

eg - H/15 * * * *, H(0-29)/10 * * * *, 45 (9-16)/2 * * 1-5
↓ every 15 min
↓ every 10 min & 1st half of every hour
↓ During 9am to 5pm every 2 hours from 9.45am on mon-fri

Page No.
Date

* Email send from jenkins -

1) configuration settings -

Manage jenkins → Configure Systems → Email Notification

SMTP server - smtp.gmail.com (different for diff. emails)

Advaced → click

☑ Use SMTP Authetication -

Username abc@gmail.com ⇒ senderemail

Password password

Use SSL ☑

SMTP port 465 (for gmail 465 & others need to search)

if want can try a test mail by adding receipient email.

→ Apply → save

2) Report email of project :-

1) Go to project

2) Configure

3) Post-build actions → Add post-build action → Email notification → Add receipient email Ids → select appropriate option check box. → Apply → save

* Custom plugins

1) Notification plugin → it allow to send status notification in JSON & XML formats

2) Extreme Notification plugin → addition facility to send WebHook (it is a http post & can be configured on any event)

3) Email-ext plugin — Advaced one for use

We customise email like when an email is sent, who should receive it & what the email says

Page No.
Date

Jeera (Jira)

Waterfall model
V-model
Agile:-
Scrum
Userstories - requirements are mentioned in it
PO, Scrum master, TL → User story assigned by any one of them
Sprint planning meeting → User story assigned in this meeting
Grooming meeting

Jira →
Jira is a project management Tool.
Jira having requirements
User story allocation is done by using jira tool.

What we do using jira -
1) User story create
2) Defect log create
3) Task create - understand user story, screenshot, expectation, create test cases & execute them
4) We can not start future sprint

Jira components
1) ~~Dashboard~~ Scrumboard Dashboard
2) Backlog ~~Deferred~~ - all newly created items (userstory, ticket) it will first appears in Backlog bucket
Afterward it will be moved as per requirement (if sprint number is not mentioned)

* Defect log -
*Defect create : priority, severity, assign to developer
Test evidence : Screenshots

Page No.
Date

* Test Report

Url →

Tenant name :

Username :

Password :

Scenario. 1. Status. Status

1. Verify Tenant Admin can able to create bill unit Passed

Screenshot

2.

* Road map in jira →

Type of project domain upcomming ?
required skill sets ?
Type of testing need to be conducted ?
Strategy plan for sprint those achievement is called roadmap

* Jira screen :-

- projectname (scrum board)
- Roadmap - milestones, Type of test } scrum master create it.
- Backlog - Red symbol - defect & green symbol - user stories
- Sprint unassigned items are available in backlog
- Active sprint - TO DO - Not started work, IN Progress - work in progress, Done - work completed
- Reports
- issues
- Components
- </> Code
- Releases
- Project pages
-

Page No.
Date

Create button →

Create issue.

Project: ▽ ⇒ List will be display for assigned project

Issue type ▽ ⇒ Epic, Task, Story, Bug

Summary

Description

Reporter: our name

priority ▽ ⇒ Highest, High, Low, Lowest

Attachment ⇒ Screenshot / testing evidence

Linked issues ▽

issue ▽ ⇒ Link with user story in which defect found

Assignee: Dev. assigned name

Epic link

Sprint: Sprint no. (active)

* Traceability matrix in excel (forward traceability)

Sr.No.	Jira ticket	User stories Description	priority	Sprint	Testcase ready
21	NAS-55	Tenant global Admin-Product scope	High	sprint-3	Yes / Not started

Estimated time in hrs	Actual time in hrs	QA status	Defect ID	Defect status
8		Failed	NA32	open

Severity	Severity of defect	QA By	Comments
	High / low	Person working from testing team	If any comment

Page No.
Date

Summary report → (in excel sheet)

1) QA sprint summary

Project	Sprint	Total user stories	Pass	Fail	Hold	in progress	Not started stories
Name of proj.	1/2/3/4	84	41	28	3	2	10

2) QA Defect summary

project	Sprint	Total Defects	Closed	open	Failed/ reopened Defect	On hold
Proj. Name	1/2/3/4	110	55	39	4	12

3) QA Test cases summary

project	Sprint	Total ~~stories~~ TC	Ready TC	In progress TC	TC NA	Not started TC
Proj. Name	1/2/3/4	84	36	00	0	48

* Task - Task/subtask is used to create to get status of work completed / todo / inprogress.

* Epic → Epic is a label used in ~~jira~~ jira so that user stories from a module/submodule can be labelled with epic name which helps in finding / tracking requirements.

If we want to manage our user story under one lebel then we create epic.

* Test case writing in excel

TC	step	step instruction	expected results	Data input	BR mapping	Pass/fail	Actual result (if failed)
TC01	Test case description is written here one by one step instructions are listed						
	1						
	2						
	3						

Defect	Business user comment

Page No.
Date

Github-Git

Github - It is a server and its services are used by many companies to optimize their code merging process.

Git - It is a tool used to push/pull code from local machine to github repository.

Check in - push

Checkout - pull

Git concept developed by founder of linux (Mr. Linus Torvalds)

* Github services -
① Wiki ② Task Management ③ bug tracking ④ Hosting services for GIT repository

* SCM - Source control management => Different types GIT, SVN, TFS (Team foundation services - microsoft)/ VSTS (Provided by microsoft)

* Download Git exe file & install
* In command promt open directory of project

Steps -

1) Initialize git into your project.
git init → initiated empty Git repository
(i.e. .git file created in that folder)

2) Link Git file with Github repository (Git add Remote origin <url>)
git remote add origin https://github.com/username/repository name.git
→ only linking is completed

Page No.
Date

Push

3) Current status (Diff. files pending to commit) (Git status)
git status → listed pending files in red colour for commit

4) Add complete project directory (Git add.)
git add . → complete project directory added

5) Repeat stage 3 to check status (Git status)
git status → Green colour

6) Commit (Git commit -m "test")
git commit -m "custom message"

7) push (Git push origin master)
git push origin master
if asked proved credentials

* for changes in files → 3+4+5+6+7
(only updated files)

Pull

1) 1st time to get repository from github to local machine
git clone url → url of project from github

2) Import from file option

for changes in files → (only updated files)
git pull origin master

Page No.
Date

GitHub Repository.

1) create new git repository
   Repositories → New.
   owner → Repository name (give some name)
   Description (if required add)
   Public / private (access level)
   Copy http link of repository to push from local repository.

2) Create local repository
   project → right click → Team → share project → (2
   select checkbox → select checkbox → create repository →
   finish → finish.

3) Eclipse to local repository –
   project → right click → Team → add to index →
   project → right click → Team → commit → add commit message
   → commit

   git status → status of file modified / new file.
   git add . → file structure will be added in bucket
   git commit -m "message any" → To add comment
   git push origin master → push in master branch
   UN, Pwd,

Page No.
Date

4) Local repository to github repository -

project → right click → Team → remote → push → enter url; enter UN & enter pwd → next → source ref → master → add specification → finish

5) Github Repository to local repository (clone repository)

In eclipse search git repository
Open git repository window → clone a git repository → enter url, enter UN & enter pwd → next → select branch & next → finish

6) Local repository to Eclipse directory.

File → import → Git → project from git → Existing local repository → select project → finish

Page No.
Date / /

Jenkin — github

Source code mgt → Git
Repository URL → copy paste from github
Credential → as per access level
Branch specifier :- */master

Build
Root pom → pom.xml
Goals & options → clean install

Log4j

log4j.jar dependency

Date

* What is log?
  - ✓ Capturing information/activities at the time of program execution
  - ✓ At execution time, jvm, selenium, testNG, all these API doing some task working together & doing their respective work
  - ✓ But we should know what is going in each & every step, what are different executions, what are diff. calls in background is going on.
  - ✓ That means we should know whats going on in execution
  - ✓ Many times execution is going on some remote machine to know execution status instead of login to remote machine we use log4j so we know execution status directly, & we can monitor log from local system
* Types of log - (level of logs)
  1) info -
  2) warn -
  3) ~~error~~ - debug
  4) fatal -
* How to generate logs?
  Using Apache log4j API
* How log4j work?
  - ✓ It reads log4j configurations from log4j.properties file
  - ✓ Log4j.properties file play important role in log report generation
  - ✓ log4j.properties file need to create in resources source folder
* src/main/resources → source folder
  log4j.properties → filename → Template available.

Page No.
Date

```
WebDriver driver;
Logger log = Logger.getLogger(LoginTest.class)
```

methods of logger

```
log.info("Type custom message"); → we at concern
log.warn("To generate warning message"); stage of action
log.fatal("To generate fatal error message");
log.debug("To generate debug message");
```

Page No.
Date

* In our framework we use :Tool - selenium & prog. lang. java.

Selenium + java

↓

Maven → Proj mgt. tool / build & manage Project

↓

TestNG - supporting framework

Src/main/java → library files
- pom classes
  - Webpage = pomclass
  - Baseclass (driver, get url, delete all cookies) ← extends
  - utility class → Screenshot, sendmail, common utility, generic fun
- Src/test/java → extends
  - Test classes (designed with TestNG)

maven dependencies → ~~pom.xml~~

pom.xml → jar download

Browser (chromeDriver, gekoDriver)

Screenshot

Test data → excel, .csv # Apache poi

Property file → env. variables / credentials

log4j jar → log generation

Extent Reporter → Report generation

Git hub repository → To store

Jenkins → for scheduling T.C. execution

Page No.
Date

Cucumber -

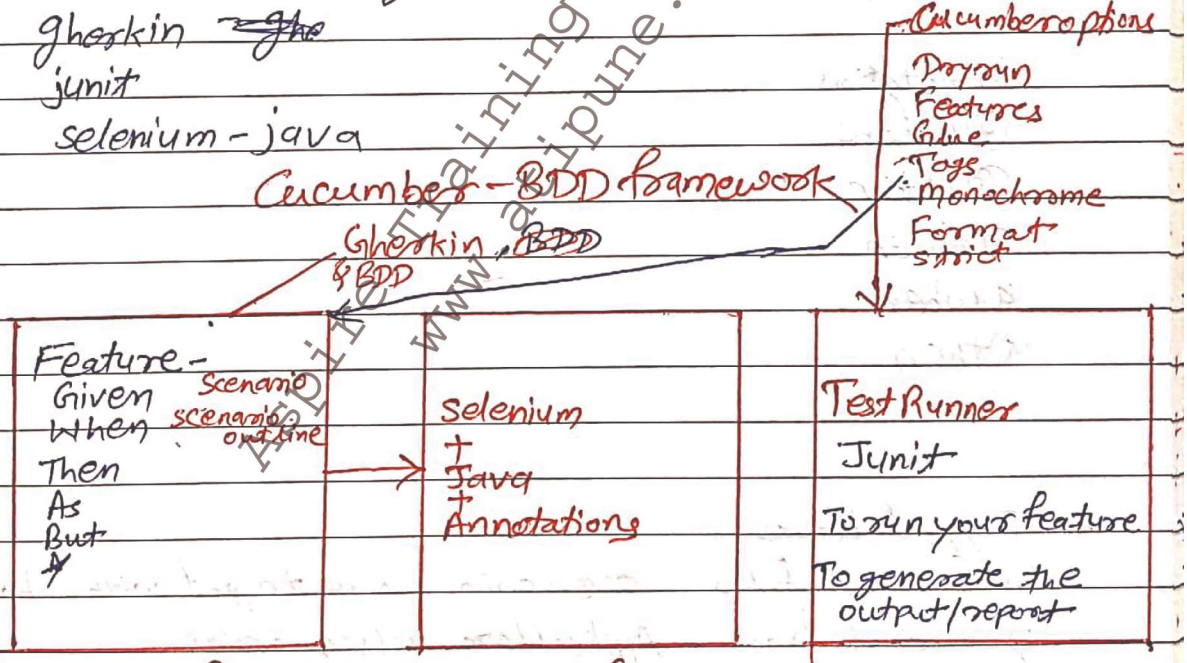
BDD framework - Behavioural Driven Development framework

Gherkin - it is a language to define BDD framework

Cucumber - JVM / Ruby → combination is used.
We use cucumber with JVM (Java)

JBehave - Java Behave

* Cucumber dependencies -
  - cucumber-java
  - cucumber-jvm
  - cucumber-junit
  - cucumber-jvm-deps
  - cucumber-reporting
  - gherkin ~~-ghe~~
  - junit
  - selenium-java



Feature file | Step definition

Package
Classes

Feature
~~login.feature~~
login.feature

Step Definations
Login Step Definations

My Runner
Test Runner

Install Natural plugin in Eclipse Marketplace

Page No.
Date

Test Runner – options in cucumber

features – feature file path

glue={"-"} path of step defination files

monochrome = true/false . Display the console output in a proper readable format

strict = true/false – it will check if any step is not defined in step definition file
(true → To display)

dry run = true/false - To check the mapping between feature file & step defination file is proper or not
(true → To check if every thing is ok)

tags = {"@SmokeTest", "@RegressionTest", "@End2End"}

In tags → OR → " , "
AND → " " ; " "
ignore → "~ "

Annotation
@Before
@After
@Given
@when
@then

* Tagged Hooks
in feature files scenarios are tagged with hooks
1) Global hook – Applicable to all scenarios
eg - browser open & close for each & every scenario

2) local hook – Applicable to concerned scenario
@Before("@first")