SmartCliff
CAREER MOBILITY SOLUTIONS

We are on a mission to address the digital skills gap for 10 Million+ young professionals, train and empower them to forge a career path into future tech

Automation Frameworks

07 APRIL, 2022

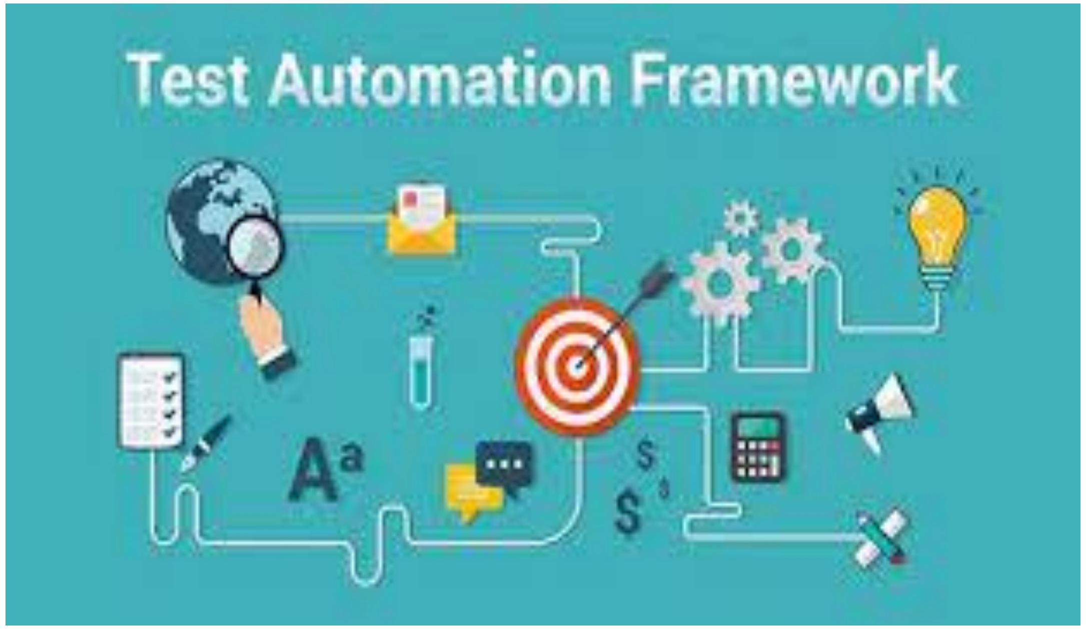
Contents

Need and Importance of Automation Framework

What is Automation Framework?

Automation Framework

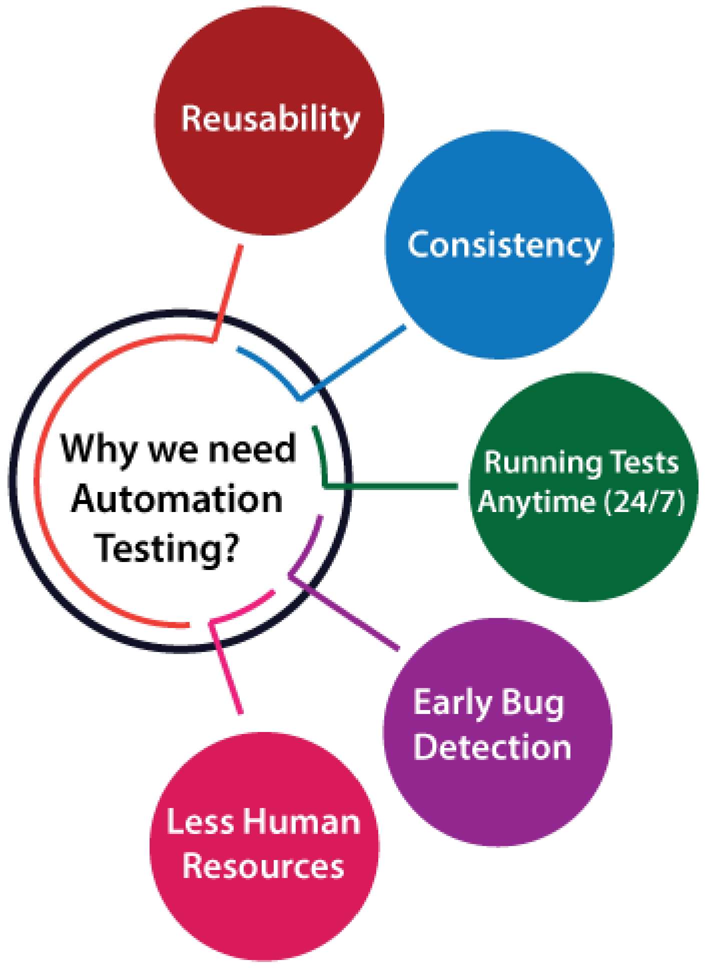
- Automation Framework is not a single tool or process, but it is a collection of tools and processes working together to support automated testing of any application.
- It integrates various functions like libraries, test data, and various reusable modules.



Why is Automation Framework?

Need of Automation Framework

- Framework supports automation testing as a technical implementation guideline.
- The automation framework not only offers the benefit of reusing the code in various scenarios, but it also helps the team to write down the test script in a standard format. Hence, the test automation framework handles all the issues.



Importance of Automation Framework

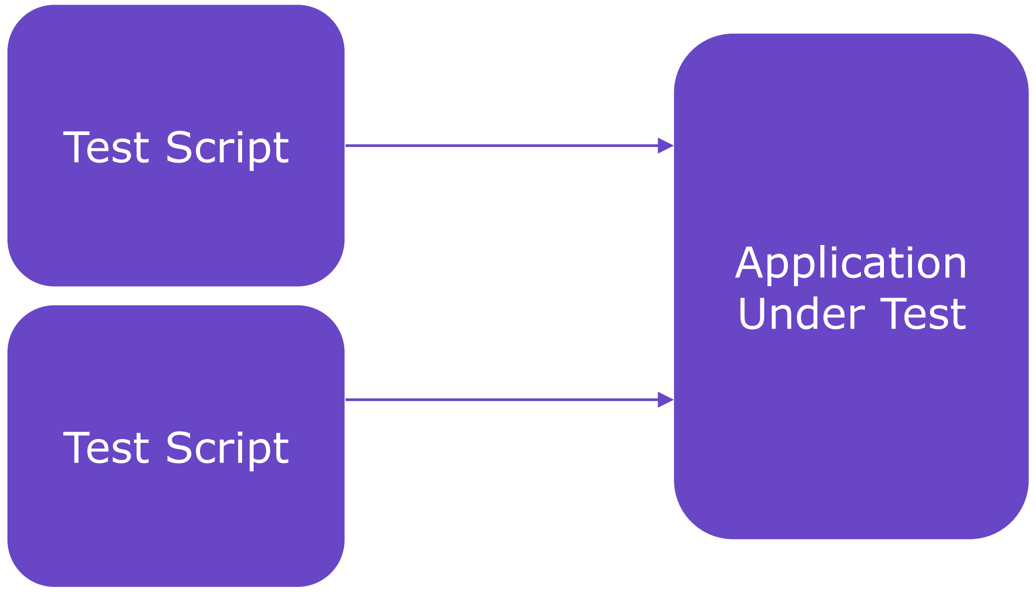
Importance of Automation Framework

- Maintain a well-defined strategy across the test suites.
- Enhance the speed at which the testing progresses.
- Maintaining the test code will be easy.
- The URL or application can be tested accurately.
- Continuous testing of code and delivery will be achieved.

Types of Automation Framework

Linear Driven Framework

- The linear Automation framework is commonly used in the testing of small applications.
- This framework is also called as a Record and playback framework.



Linear Driven Framework (Contd.)

Pros

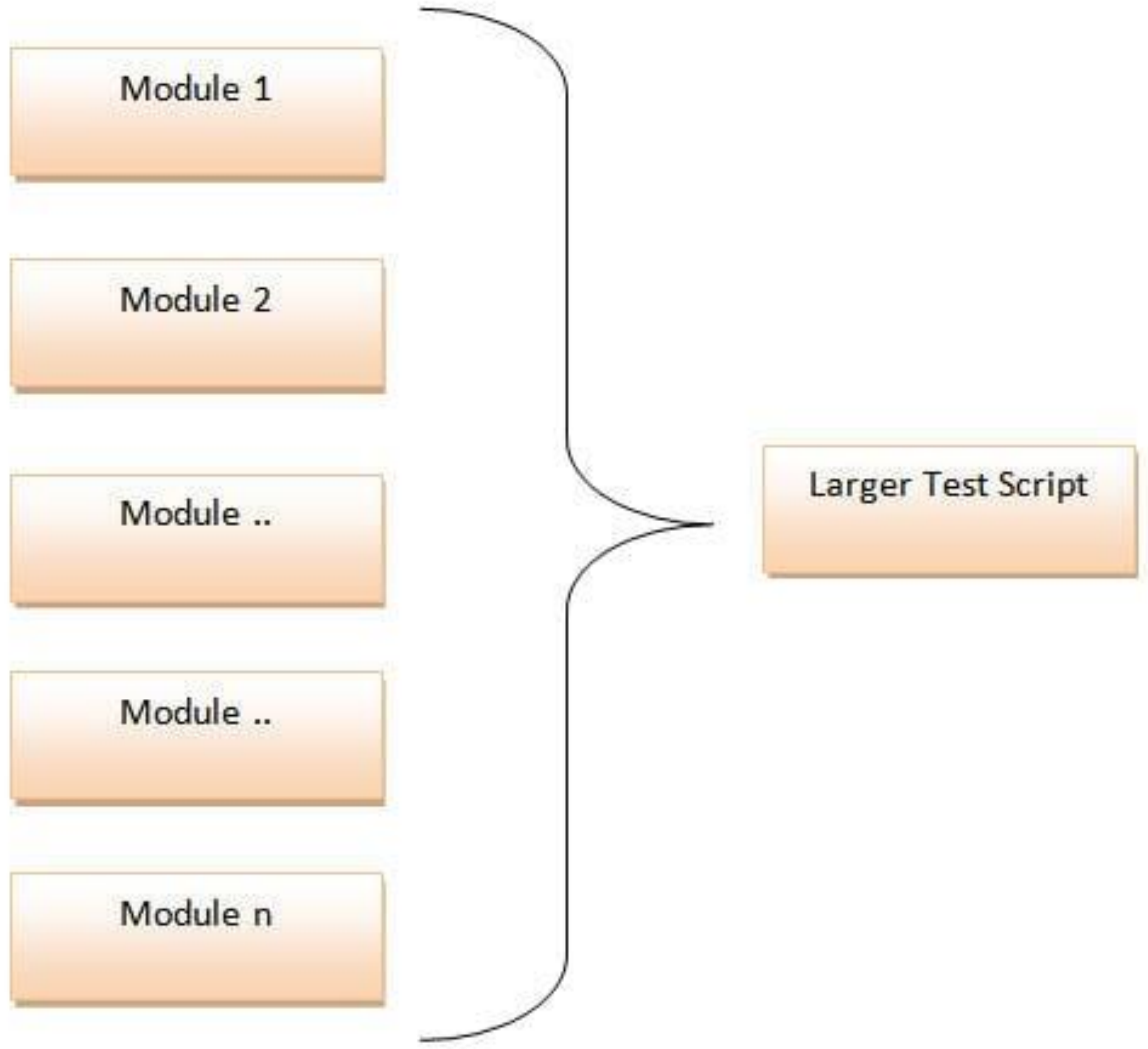
- There is no need to write custom code, so expertise in test automation is not necessary

Cons

- The data is hardcoded in the test script; hence, the test cases cannot be re-run with multiple sets. You need to make some changes if the data is altered.

Modular Driven Framework

- In this Framework, the tester can create test scripts module wise by breaking down the whole application into smaller modules as per the client requirements and create test scripts individually



Modular Driven Framework (Contd.)

Pros

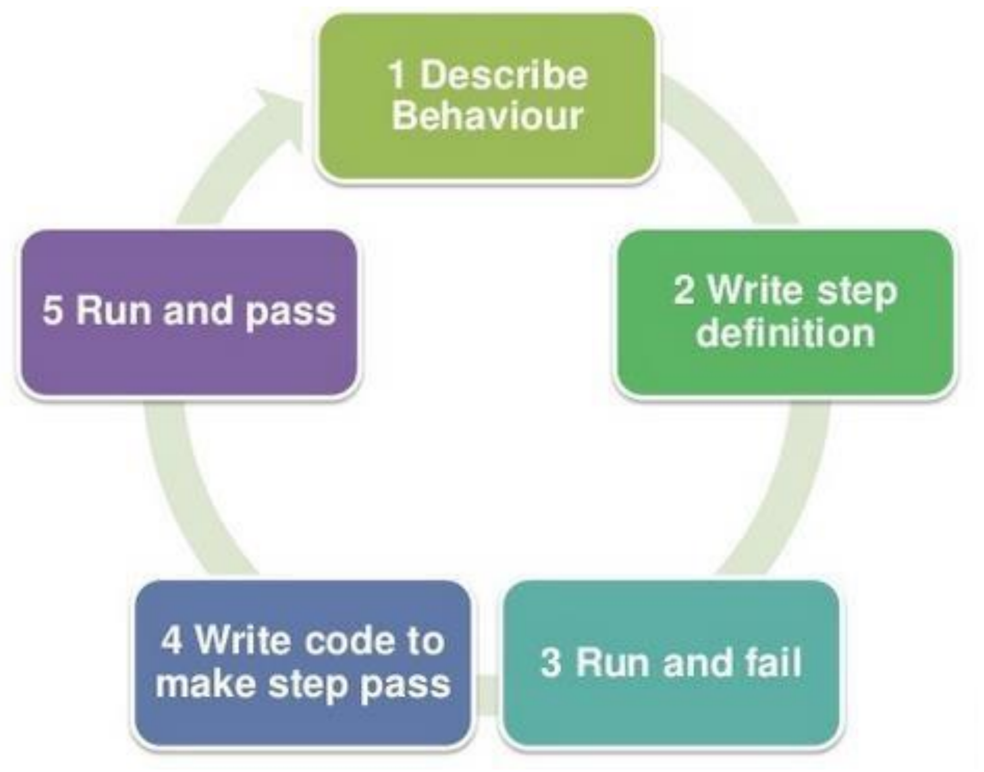
- Modular driven framework ensures the division of scripts that leads to easier maintenance and scalability. You can write independent test scripts.

Cons

- The modular driven framework requires additional time in analyzing the test cases and identifying reusable flows.

Behavior Driven Framework

- Behavior Driven Development framework is to create a platform, which allows every person, like Developers, Testers, business analyst, etc., to participate actively.
- It also increases collaboration between the tester and the developers on your project.



Behavior Driven Framework (Contd.)

Pros

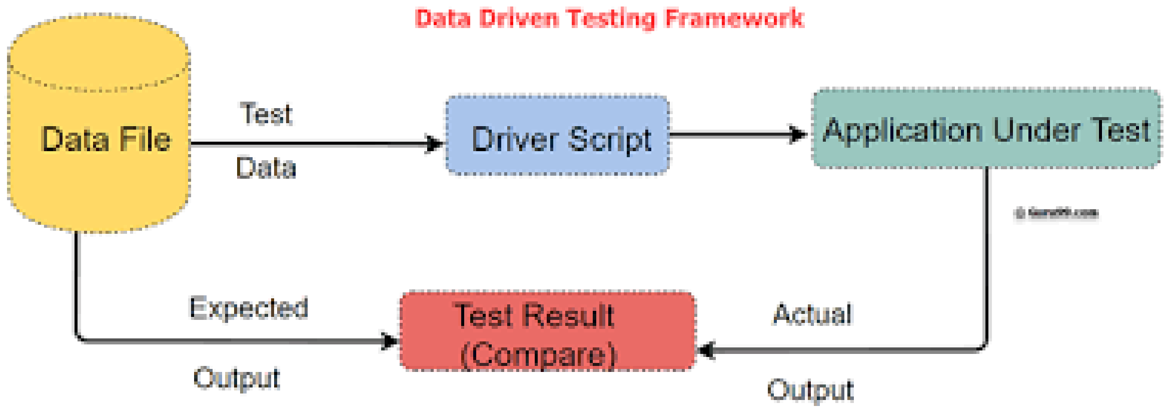
- We can user non-technical, natural language to create test specifications on this behavior-driven testing.

Cons

- To work with this framework, sufficient technical skills as well as prior experience in Test driven development is required.

Data Driven Framework

- Generally, Test Data is read from the external files like Excel Files, Text Files, CSV Files, ODBC Sources, DAO Objects and they are loaded into the variables inside the Test Script.
- The data-driven framework allows us to create test automation scripts by passing different sets of test data.



Data Driven Framework (Contd.)

Pros

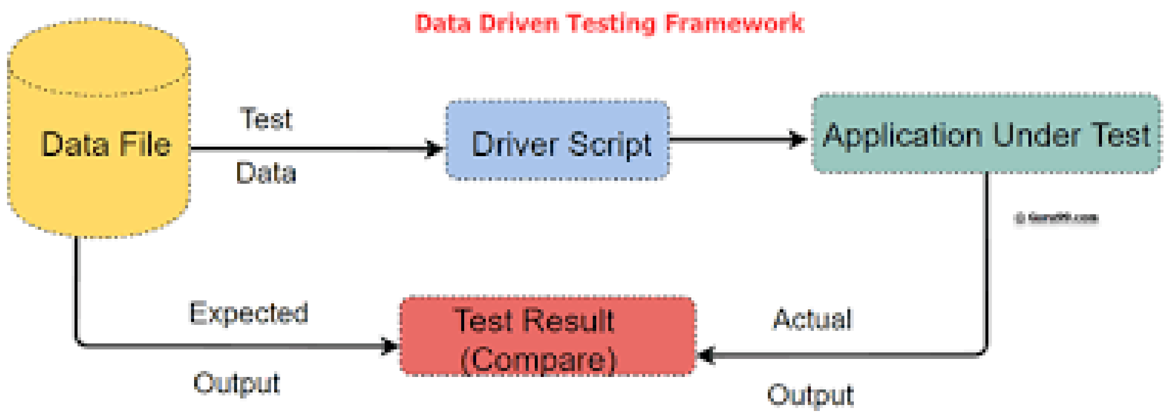
- It reduces the number of scripts required. Hence, multiple scenarios can be tested in less code.

Cons

- You will need a highly experienced tester who should be proficient in various programming languages to completely utilize the design of this framework.

Keyword Driven Framework

- The Keyword-Driven Testing framework is also known as a table-driven testing.
- This framework is suitable only for small projects or applications
- The automation test scripts performed are based on the keywords specified in the excel sheet of the project.



Keyword Driven Framework (Contd.)

Pros

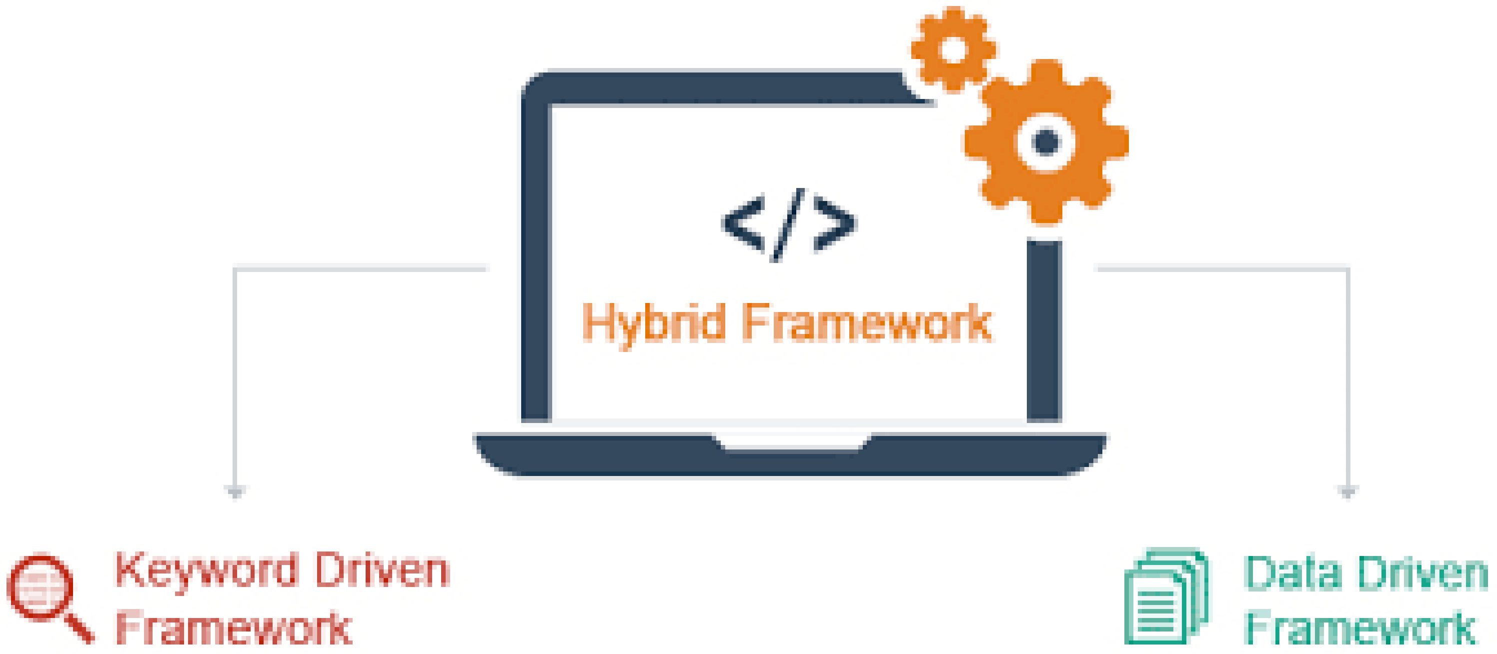
- A single keyword can be used across multiple test scripts, so the code is reusable.

Cons

- The initial cost of setting up the framework is high, and it is time consuming & complex.

Hybrid Framework

- Hybrid Framework is used to combine the benefits of Keyword Driven and Data-Driven Frameworks



Hybrid Framework (Contd.)

Pros

- This type leverages the advantages of all kinds of related frameworks.

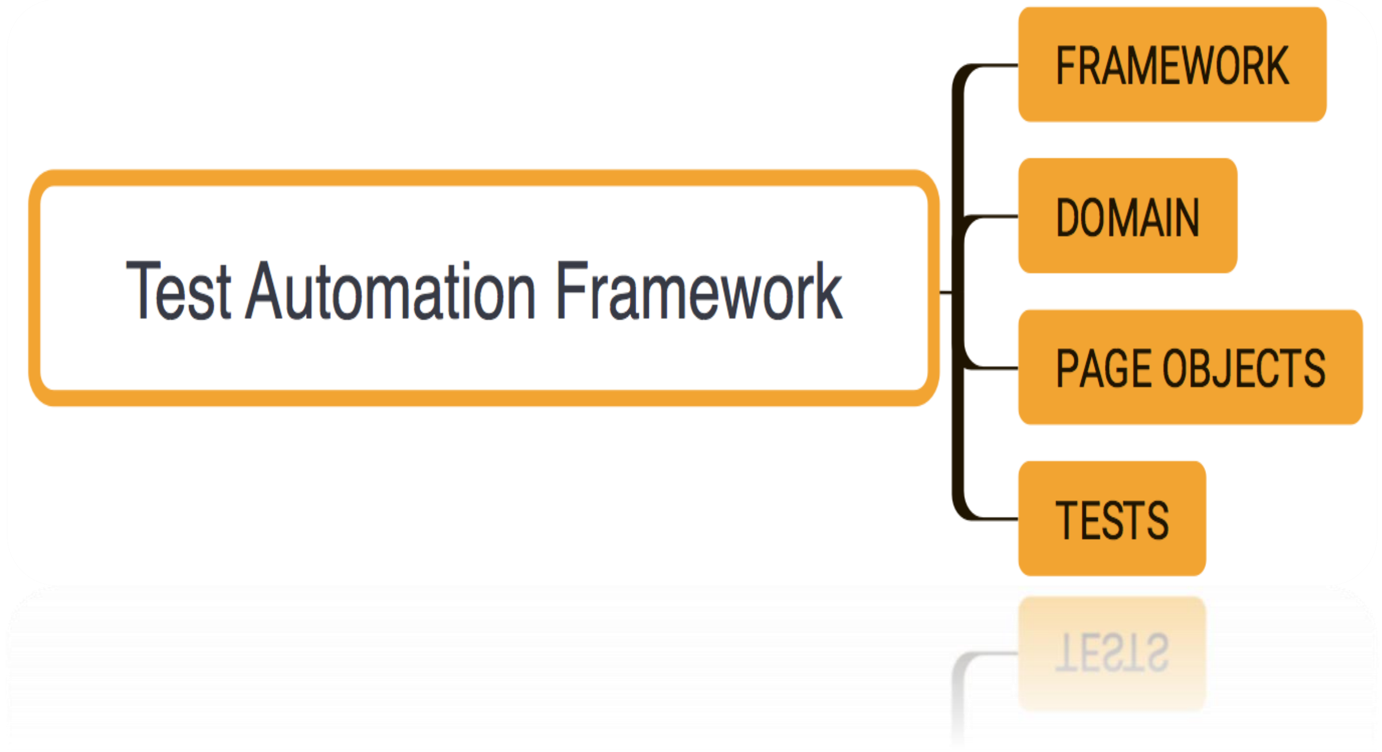
Cons

- Tests are fully scripted in Hybrid Testing framework thus increase the automation effort

Automation Framework Design

Test Automation Frameworks

Frameworks



How to build a maintainable Selenium framework?

- Choose a programming language
- Choose a unit test framework
- Design the framework architecture
- Build the selenium core component
- Build the selenium test component
- Choose the reporting mechanism
- Decide how to implement CI/CD
- Integrate your framework with other tools

How Do You Choose a Programming Language?

Usability

Command-Line Strength

Simplicity

IDE Support

Test Framework Support

Industry Adoption

Choosing a Unit Test Framework

- Modular - The framework should be adaptable to change.
- Reusable – The commonly used methods or utilities should be kept in a common file which is easily accessible to all the scripts.
- Consistent – The test suite should be written in a consistent format by following the coding practices.
- Independent – The test scripts should be written in such a way that they are independent of each other.

Designing the Architecture of your Framework

1) Structure, Organize & Set Up Source Control
2) Familiarize yourself with the Application
3) Determine your testing environments & Gather Data
4) Set up a smoke test project
5) Create Utilities for on screen actions
6) Build and Manage Verifications
7) Set up your logging & reporting mechanism

Choose a Mechanism for Reporting

Reporting

- Unless we have a great reporting mechanism, it would not be easy to read the test results
- We would need to convert test results into insights, which can produce corrective results instantly.
- We can find a number of options to help to log your automated tests.
- In order to report this, We can use testing frameworks, including TestNG and JUnit.
- These are not in human-readable and need to be accessed using software such as CI or CD servers.

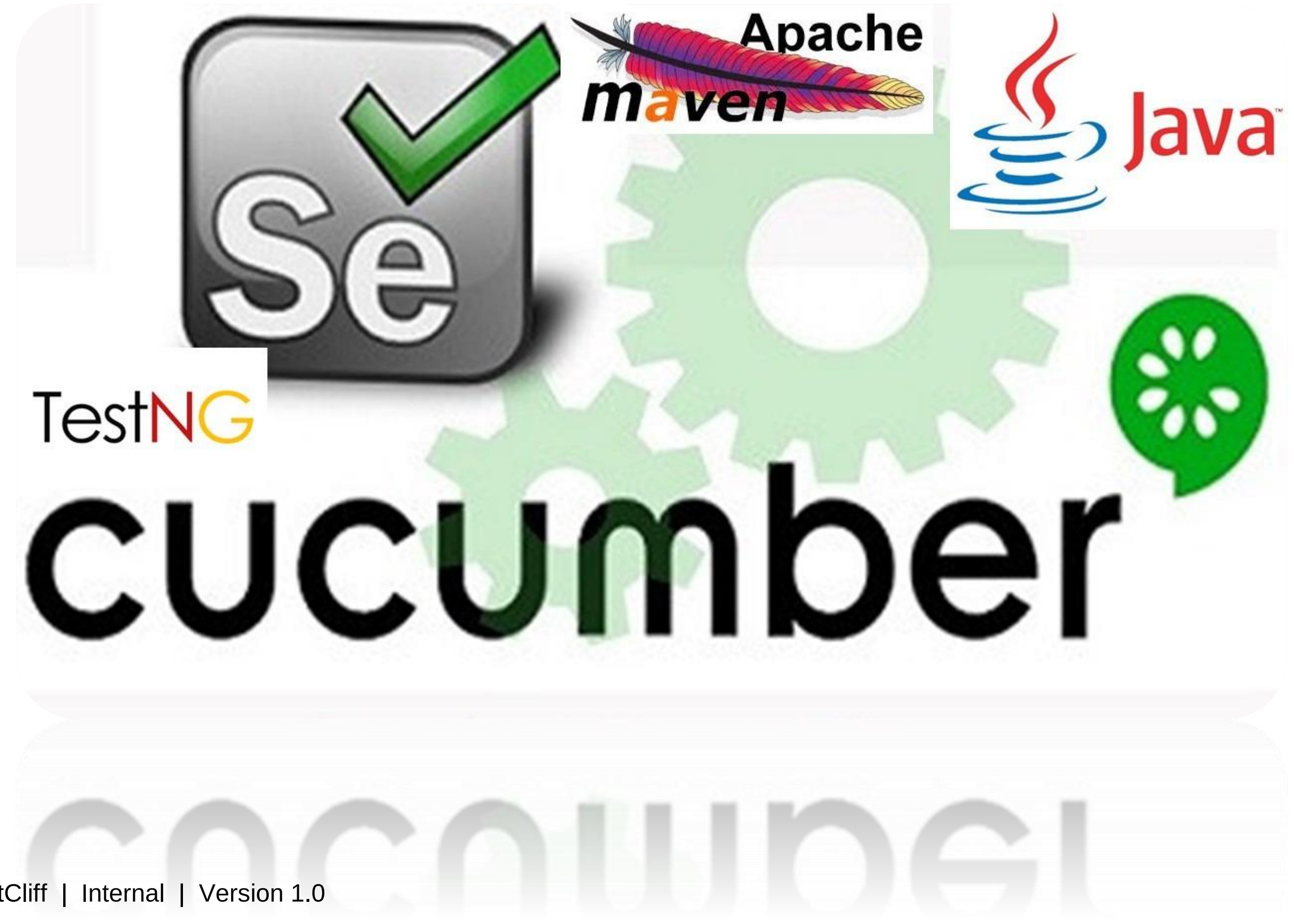
Choose a Mechanism for Reporting (Contd.)

Reporting

- But if we are using third-party libraries, including Allure or Extent Report, you would be able to create test result reports, which are in human-readable forms.
- Also, these would include screenshots and pie charts.
- These come with certain open-source reporting libraries, including Report NG for Java, which is an HTML reporting plugin for the TestNG framework.

Cucumber Framework

CUCUMBER Framework



What is Cucumber?

Introduction

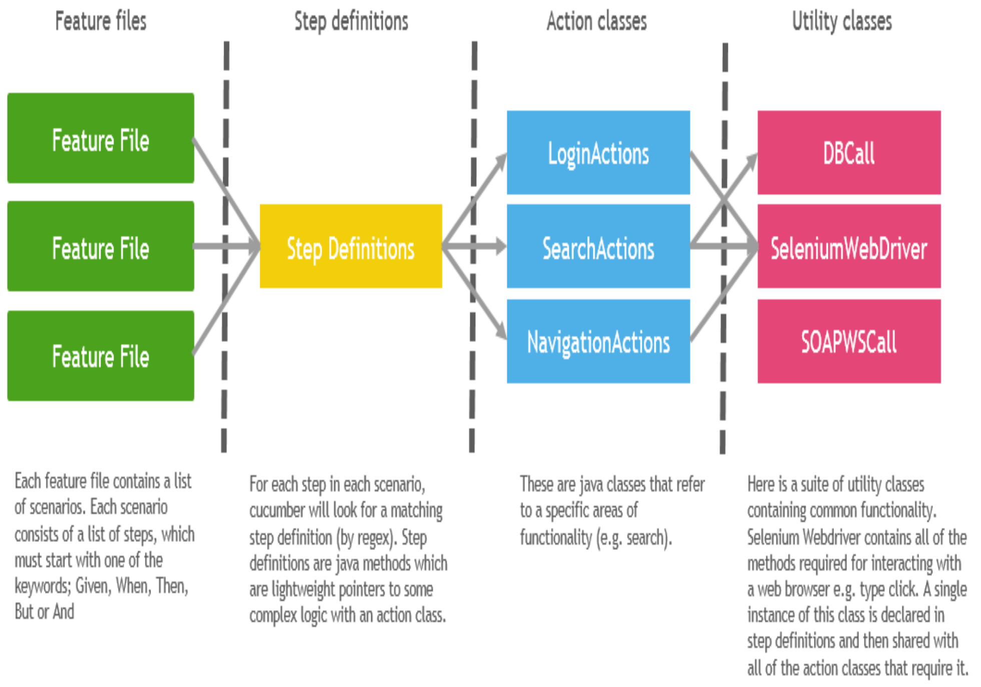
- Cucumber is a *software tool used by the testers to develop test cases for the testing of behavior of the software.*
- It plays a supporting role in automated testing.
- It is used in the development of acceptance test cases for web application automation as per the behavior of their functionalities.
- In the Cucumber testing, the test cases are written in a simple English text, which anybody can understand without technical expertise.

What is Cucumber? (Contd.)

- This simple English text is called the Gherkin language.
- It allows business analysts, developers, testers, etc. to automate functional verification and validation in an easily readable and understandable format (e.g., plain English).

What is Gherkin?

- Gherkin is a business readable language which helps you to describe business behavior without going into details of implementation.
- The text in Gherkin language acts as documentation and skeleton of your automated tests.
- Gherkin is line-oriented language
- Each line called step and starts with keyword and end of the terminals with a stop



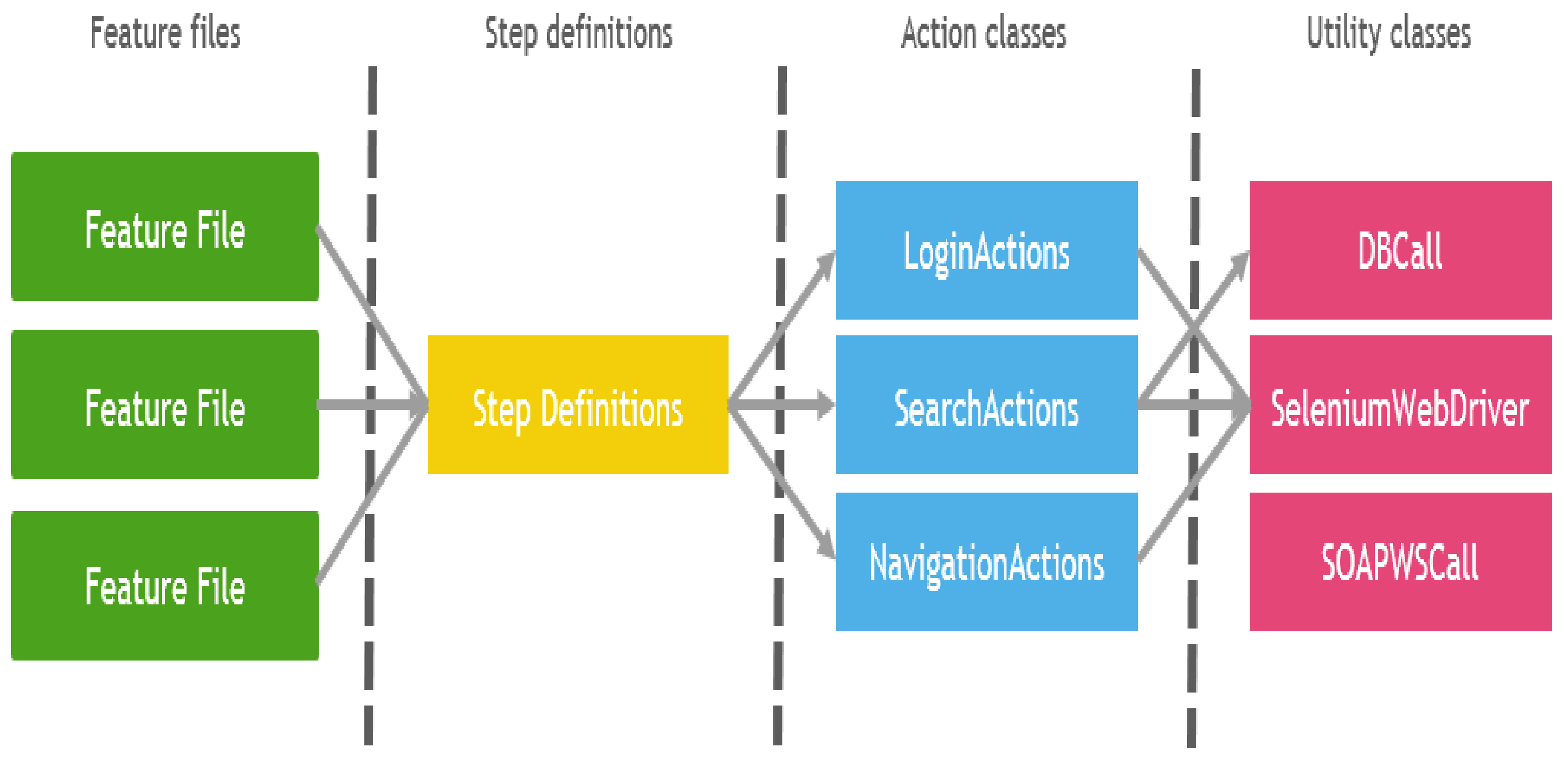
Each feature file contains a list of scenarios. Each scenario consists of a list of steps, which must start with one of the keywords; Given, When, Then, But or And

For each step in each scenario, cucumber will look for a matching step definition (by regex). Step definitions are java methods which are lightweight pointers to some complex logic with an action class.

These are java classes that refer to a specific areas of functionality (e.g. search).

Here is a suite of utility classes containing common functionality. Selenium Webdriver contains all of the methods required for interacting with a web browser e.g. type click. A single instance of this class is declared in step definitions and then shared with all of the action classes that require it.

What is Gherkin?



Each feature file contains a list of scenarios. Each scenario consists of a list of steps, which must start with one of the keywords; Given, When, Then, But or And

For each step in each scenario, cucumber will look for a matching step definition (by regex). Step definitions are java methods which are lightweight pointers to some complex logic with an action class.

These are java classes that refer to a specific areas of functionality (e.g. search).

Here is a suite of utility classes containing common functionality. Selenium Webdriver contains all of the methods required for interacting with a web browser e.g. type click. A single instance of this class is declared in step definitions and then shared with all of the action classes that require it.

What is Gherkin? (Contd.)

- In this script, a comment can be added anywhere you want, but it should start with a # sign

Syntax of Gherkin :

Feature: Title of the Feature

Scenario: Title of the Scenario

Given [Preconditions or Initial Context]

When [Event or Trigger]

Then [Expected output]

- A Gherkin document has an extension .feature

What is Gherkin? (Contd.)

- Cucumber reads Gherkin document and executes a test to validate that the software behaves as per the Gherkin syntax.

Important Terms used in Gherkin:

1) Feature
2) Background
3) Scenario
4) Given
5) When

What is Gherkin? (Contd.)

6) Then

7) And

8) But

9) Scenario Outline Examples

Feature :

- **The file should have extension .feature** and *each feature file should have only one feature*.
- The feature keyword being with the **Feature:** and after that add, a space and name of the feature will be written.

What is Gherkin? (Contd.)

Scenario:

- **Each feature file may have multiple scenarios,** and each scenario starts with Scenario: followed by scenario name.

Background:

- Background keyword helps you to **add some context to the scenario**. It can contain some steps of the scenario, but the only difference is that it should be run before each scenario.

What is Gherkin? (Contd.)

Given:

- The use of Given keyword is to put the system in a familiar state before the user starts interacting with the system.

Syntax:

Given - a test step that defines the 'context

Given I am on "/."a

What is Gherkin? (Contd.)

When:

- When- step is to define action performed by the user.
- **Syntax:**

A When - a test step that defines the 'action' performed

When I perform "Sign In."

What is Gherkin? (Contd.)

Then:

- The use of ‘then’ keyword is to see the outcome after the action in when step. However, you can only verify noticeable changes.

Syntax:

Then - test step that defines the 'outcome.'

Then I should see "Welcome Tom."

What is Gherkin? (Contd.)

And & But:

- You may have multiple given when or Then.

Syntax:

But - additional test step which defines the 'action' 'outcome.'

But I should see "Welcome Tom.“

Syntax:

And - additional test step that defines the 'action' performed

And I write "EmailAddress" with "Tomjohn@gmail.com."

What is Gherkin? (Contd.)

Example:

- **Feature**: Login functionality of social networking site Facebook.
- **Given** I am a facebook user.
- **When** I enter username as username.
- **And** I enter the password as the password
- **Then** I should be redirected to the home page of facebook

- The scenario mentioned above is of a feature called user login.
- All the words written in bold are Gherkin keywords.

What is Gherkin? (Contd.)

- Gherkin will analyze each step written in the step definition file.
- Therefore, the steps are given in the feature file and the step definition file should match.

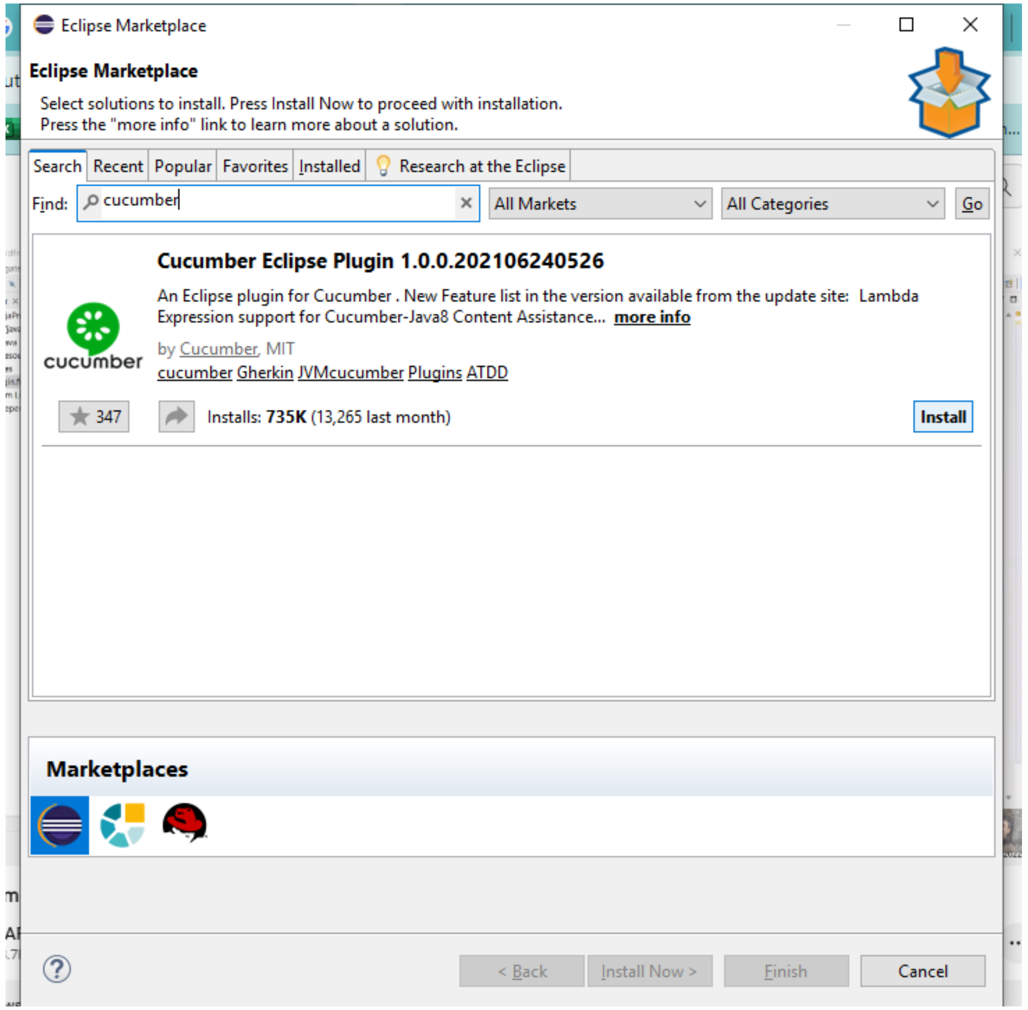
Summary:

- Gherkin is the format for cucumber specifications
- Gherkin is line-oriented language just like YAML and Python
- Gherkin Scripts connects the human concept of cause and effect to the software concept of input/process and output

What is Gherkin? (Contd.)

- Feature, Background, Scenario, Given, When, Then, And But are importantly used in Gherkin
- In Gherkin, each scenario should execute separately
- The biggest advantage of Gherkin is simple enough for non-programmers to understand
- Gherkin Test may not work well in all type of scenarios

Install cucumber eclipse plugin from Eclipse market place



Eclipse Marketplace

Eclipse Marketplace

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

Find: cucumber | All Markets | All Categories | Go

Cucumber Eclipse Plugin 1.0.0.202106240526

An Eclipse plugin for Cucumber . New Feature list in the version available from the update site: Lambda Expression support for Cucumber-Java8 Content Assistance... **more info**

by Cucumber, MIT

cucumber Gherkin JVMcucumber Plugins ATDD

347 Installs: **735K** (13,265 last month) **Install**

Marketplaces

Dependency to be added for cucumber

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
<groupId>io.cucumber</groupId>
<artifactId>cucumber-java</artifactId>
<version>7.13.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>4.12.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
<dependency>
<groupId>io.github.bonigarcia</groupId>
<artifactId>webdrivermanager</artifactId>
<version>5.5.3</version>
</dependency>
```

Dependency to be added for cucumber (contd..)

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
<groupId>io.cucumber</groupId>
<artifactId>cucumber-junit</artifactId>
<version>7.13.0</version>
<scope>test</scope>
</dependency>
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
<groupId>org.testng</groupId>
<artifactId>testng</artifactId>
<version>7.8.0</version>
<scope>test</scope>
</dependency>
```

Writing Test Cases Using Cucumber

Cucumber includes the following three files:

1.Feature file: Here we write the Features to be tested in Gherkin format i.e. Given When Then. We can even run the feature file to execute the test scripts written in the Stepdef file.

2.Stepdef file: Once the Feature file is ready, each sentence of the Feature file can be further implemented over the Stepdef file.

3.Runner File: This is just to execute the actual test script written over the Stepdef file by referring to the feature file. Apart from that, it has many other options for customization, reporting, selective execution, etc.

Here is a simple example of a Runner File

```
package runner;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
features="Features",
glue="com.Multi",
plugin={"html:target/cucumber-html-report", "json:target/cucumber.json",

"pretty:target/cucumber-pretty.txt","usage:target/cucumber-usage.json", "junit:target/cucumber-results.xml"},
dryRun = false,
monochrome = true,
tags={"@Smoke,@Regression"}
)

public class Runner {

}
```

Creating Files For Cucumber

- **Stepdef file**– Src/test/java >> New >> Others >> Cucumber>>StepDef class.
- **Feature file**– Project>>New>>File>> specify name for the file with extension as ‘.feature’.
- **Runner file**– It is similar to any other Java class creation but we may require implementing some method here.

Cucumber Features

#1) Cucumber Hooks

These are the blocks of the code that runs before or after each scenario. So that we can define these, anywhere in our project. Example, Step Definition.

As per the definition, these are just two annotation @After and @Before. In the console, we can see the blocks getting executed and giving clear output. We can also execute the hooks for specific Tags.

#2) Cucumber Tags

These are normally used over the feature file to classify the scenarios over the feature files as per their given tag name. We can have multiple tags for a given scenario in the feature file.

Tags are user-defined and we can give any name to it such as @Smoke, @Regression, etc.

Cucumber Features

#3) Cucumber Annotations

These are inbuilt to Cucumber. Normally tags are @Given, @When, @Then.

However, later if we need we can create our own annotation and then use it in our program. During execution, the matching glue code i.e. functions are written in a Stepdef file having @Given, @When, @Then will get executed.

#4) Cucumber Background

These are steps or series of steps that are common to all the scenarios in the feature file.

It allows us to add some context to the scenarios for a feature where it is defined. It runs before every scenario for a feature in which it is defined.

#5) Cucumber Data Tables

Cucumber has the feature to support data-driven testing, which allows us to automatically run a test case multiple times with different input and validation values for a given script.

Cucumber supports the data table. The first row is considered as the column and the rows next to it are the data for the scripts.

#6) Cucumber Transpose

This is a slight modification to the Cucumber data table. Here the first column would be considered as column and the columns next are considered as data for the scripts.

#7) Cucumber Multi Scenarios

Cucumber allows us to perform testing multiple scenarios under one feature file.

#8) Cucumber Reporting

Unlike reporting and other third-party tools where we need to do some configuration to view the reporting.

#9) Scenario Outline (incase of multiple scenarios)

Scenario outlines are used when the same test is performed multiple times with a different combination of values

Cucumber Hooks and tags example

```
public class Hooksexample1 {
WebDriver driver;
@Before("@T1")
public void before()
{System.out.println("Before test");
}
@Given("Browser opens")
public void browser_opens() {
WebDriverManager.chromedriver().setup();
driver=new ChromeDriver();
}
@When("open google page")
public void open_google_page() {
driver.get("https://google.com");
String expectedString="Google";
}
```

```
@Then("validates title of the page with expected value")
public void validates_title_of_the_page_with_expected_value() {
Assert.assertEquals("Google", driver.getTitle());
}
@After
public void after()
{
System.out.println("After test");
}
}
```

Feature file

Feature: Google Title Search

```
@T1
Scenario: Test the title of the Googlepage
Given Browser opens
When open google page
Then validates title of the page with expected value
```

TestNG With Cucumber

We can still execute the JUnit test cases written in Cucumber using TestNG by following**:**

1.We need to add the dependencies to the Maven project.

2.Extend the class in Runner class as AbstractTestNGCucumberTests package runner.

3.Convert the maven project and add the package(where the runner class exists).

After that, we can run the entire Cucumber test case as TestNG and generate reports relating to the same(if we have the listeners).

Once the Test Environment is setup:

1. Add Eclipse Cucumber plugin in Eclipse.

2. Create a Maven project and add all the required dependencies to it and also add TestNG related dependency in Maven.

3. Create a new feature file.

4. Add the required implementation for it in the Stepdef file.

5. Now create a runner file with extends AbstractTestNGCucumberTests.

6. Convert the Maven project to TestNG and in the testng.xml add the package path of Cucumber runner class.
7. Run the TestNG.xml file.

Feature file

Feature:
As a user
I want to be able to add new clients in the system
So that I can add accounting data for that client

Background:
Given I am on Github home page
When I specify Username as “xxxxxxxxxxxxxxxxxxxx” and Password as “xxx”
And Click on SignIn button

@Smoke
  Scenario: Editing the profile
Given I click on Your Profile option
When I click on edit profile button
And Uploaded new picture
Then I should be seeing new profile picture

@Regression @Everytime
  Scenario: Create new gist
Given I click on Your Gists option
When I provide filename, description
And click on Create public gist method
Then I should be seeing the new gist

Stepdef File

```
public static void myanno()
{
	System.out.println("my annot gets executed");
}
    @Given("^I am on Github home page$")
    public void i_am_on_Github_home_page(){
        String site = "https://www.github.com/login";
        System.setProperty("webdriver.chrome.driver",
"Executables\\chromedriver.exe");
        driver = new ChromeDriver();
        driver.navigate().to(site);

//driver.findElement(By.cssSelector("a[contains[@href,
"login"]]").click();
        // Write code here that turns the phrase above into
concrete actions
    }
```

```
    @When("^I specify Username as \"([^\"]*)\" and Password as
\"([^\"]*)\"$")
    public void i_specify_Username_as_and_Password_as(String
arg1, String arg2){

driver.findElement(By.cssSelector("input#login_field")).sendKeys(a
rg1);

driver.findElement(By.cssSelector("input#password")).sendKeys(a
rg2);
        // Write code here that turns the phrase above into concrete
actions
    }
    @When("^Click on SignIn button$")
    public void click_on_SignIn_button(){
        driver.findElement(By.cssSelector("input.btn")).click();
        // Write code here that turns the phrase above into concrete
actions
    }
```

```
@Given("^I click on Your Profile option$")
    public void i_click_on_Your_Profile_option(){
        driver.findElement(By.xpath("//*[@id='user
-links']/li[3]/details/summary/img")).click();
        List<WebElement> olist = driver.findElements(By.xpath("//*[@id=
'user-links']/li[3]/details/ul/li/a[@class='dropdown-item']"));
        for(WebElement o:olist)
        {
            if(o.getText().equals("Your profile"))
            {
                o.click();
                break;
            }
        }
```

```
    @When("^Uploaded new picture$")
  public void uploaded_new_picture() throws InterruptedException{
        WebElement s1 =
driver.findElement(By.xpath("//*[@class='avatar-upload
-container clearfix']/Img"));
        sb=s1.getAttribute("src");
        System.out.println(s1.getAttribute("src"));
        driver.findElement(By.id("upload-profile
-picture")).sendKeys("D://cucumberFinal//multiple//Files//images.jpg"
        Thread.sleep(10000);
        String wh = driver.getWindowHandle();
        driver.switchTo().window(wh);
// Write code here that turns the phrase above into concrete actions
    }
    @When("^I click on edit profile button$")
    public void i_click_on_edit_profile_button(){
        driver.findElement(By.xpath("//*[@id='js-pjax
-container']/div/div[2]/div[1]/a")).click();
        // Write code here that turns the phrase above into concrete
actions
    }
```

data tables and transpose

Feature File

Feature: Title of your feature
I want to use this template for my feature file

Background:
Given I am on Gmail login page
When I specify Username and Password
And Click on SignIn button

Scenario: Create new message from data table
When I am on New Email Page
And I specify the following details
To1	Subject
Person1@email.com	Person1 subject
Person2@email.com	Person2 subject

Scenario: Create new message from transposed data table
When I am on New Email Page
And I specify following details from transpose table
| To1 | Person1@email.com | Person2@email.com |
| Subject | Person1 subject | Person2 subject |

Stepdef file

```
public class Step3 {
    static public WebDriver driver;
    @Given("^I am on Gmail login page$")
    public void i_am_on_Gmail_login_page() throws InterruptedException{
        System.setProperty("webdriver.chrome.driver",
 "Executables/chromedriver.exe");
        driver= new ChromeDriver();
        Actions act = new Actions(driver);
        driver.manage().window().maximize();
        driver.navigate().to("https://www.google.com/gmail/about");
        driver.findElement(By.xpath("//nav/div/a[2]")).click();
        Thread.sleep(3000);
    }
    @When("^I specify Username and Password$")
    public void i_specify_Username_and_Password() throws InterruptedException{
        driver.findElement(By.xpath("//input[@type='email']")).sendKeys("xxxxxx@xxx.
com");
```

Stepdef file

```
driver.findElement(By.xpath("//*[@id='identifierNext']/content/span")).click();
        Thread.sleep(3000);
        driver.findElement(By.xpath("//input[@type='password']")).sendKeys("xxxxxxxx
xxx");
    }

    @When("^Click on SignIn button$")
    public void click_on_SignIn_button() throws InterruptedException{

        driver.findElement(By.xpath("//*[@id='passwordNext']/content/span")).click();
       Thread.sleep(5000);
    }
```

```
@When("^I am on New Email Page$")
    public void i_am_on_New_Email_Page(){
    }
    @When("^I specify following details$")
    public void i_specify_following_details(DataTable tables)throws
Throwable{
            for (Map<String, String> row : tables.asMaps(String.class,
 String.class))
            {
                driver.findElement(By.xpath("//*[@id=':x4']/div/div")).click();
                //driver.switchTo().
                System.out.println(row.get("To1"));
                System.out.println(row.get("Subject"));
                String whandle = driver.getWindowHandle();
                driver.switchTo().window(whandle);
driver.findElement(By.xpath("//td[@class='eV']/div[1]/div/textarea")).se
ndKeys
(row.get("To1"));
                driver.findElement(By.xpath("//table[@class='aoP aoC
 bvf']/tbody/tr/td/form/div[3]/input")).sendKeys(row.get("Subject"));

                driver.findElement(By.xpath("//table[@class='IZ']/tbody/tr/td/
div")).click();
           Thread.sleep(3000);
            }
         }
```

Program 1 (Contd.)

```
public class OpenExcel {
String[][] data=null;
@DataProvider(name="Exceldata")
public String[][] data1() throws BiffException, IOException
{
data=readExcel();
System.out.println(data);
return data;
    }
String[][] readExcel() throws BiffException, IOException
{
FileInputStream file=new FileInputStream("c:\\SWARNA\\credentials1.xls");
Workbook workbook=Workbook.getWorkbook(file);
Sheet sheet=workbook.getSheet(0);
```

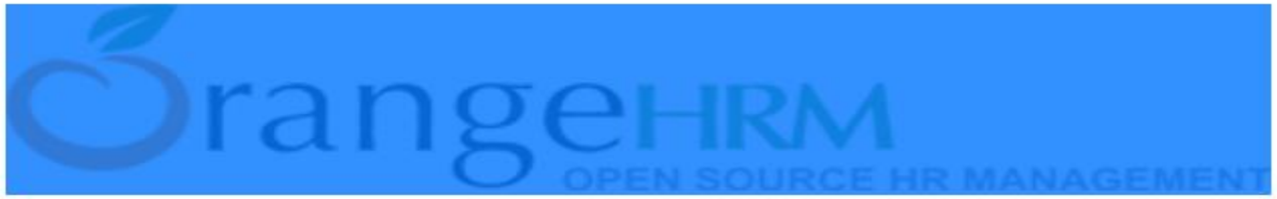
Program 1

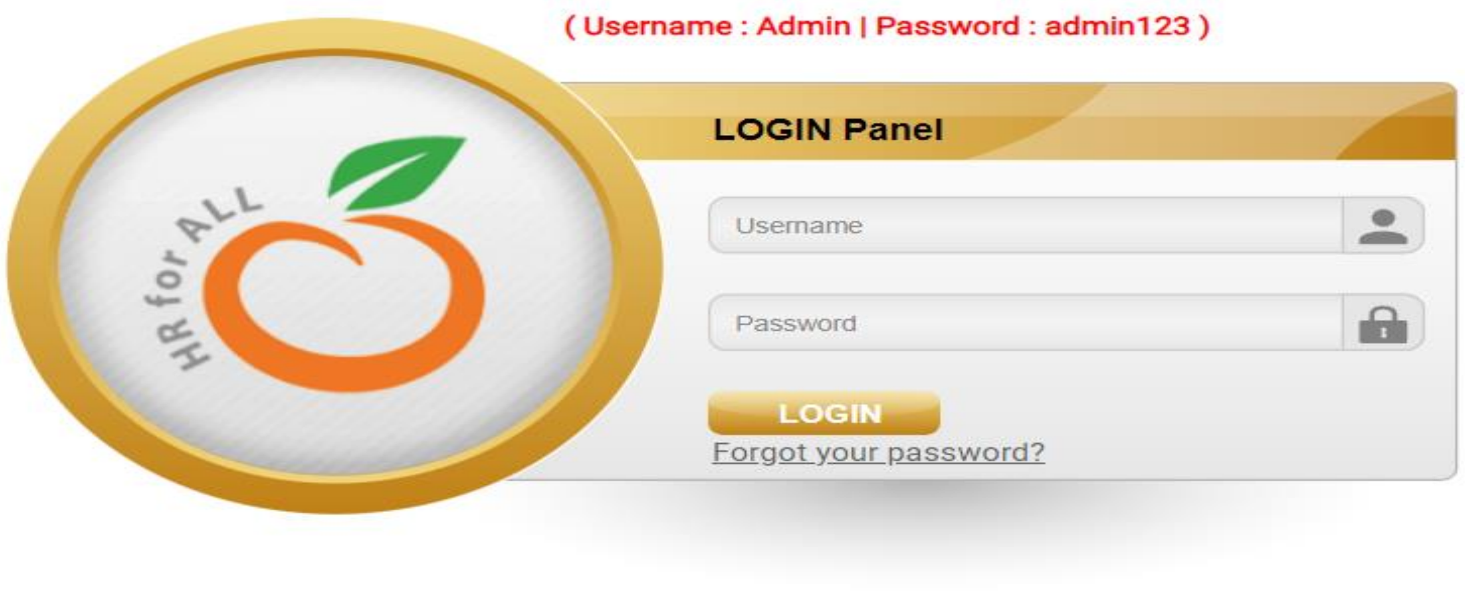
```
int rowcount=sheet.getRows();
int colcount=sheet.getColumns();
String[][] data1=new String[rowcount][colcount];
for(int i=1;i<rowcount;i++)
{
for(int j=0;j<colcount;j++)
{
data1[i-1][j]=sheet.getCell(j,i).getContents();
}
}
return data1;
}
```

Program 1 (Contd.)

```
@Test(dataProvider="Exceldata")
public void login_all(String user,String passwd)
{
System.setProperty("webdriver.chrome.driver","c:\\Drivers\\chromedriver.exe");
WebDriver driver=new ChromeDriver();
driver.get("https://opensource-demo.orangehrmlive.com/index.php/validatecredentials");
WebElement userName=driver.findElement(By.id("txtUsername"));
WebElement  password=driver.findElement(By.id("txtPassword"));
WebElement loginButton=driver.findElement(By.id("btnLogin"));
userName.sendKeys(user);
password.sendKeys(passwd);
loginButton.click();
driver.quit();
}}
```

Output




(Username : Admin | Password : admin123)

Program 2

```
public class Usingpoi {
@Test
public void read() throws IOException
{
FileInputStream file=new
FileInputStream("C:\\Users\\windlows\\Downloads\\LoginDaTa.xls");
HSSFWorkbook workbook=new HSSFWorkbook(file);
HSSFSheet sheet=workbook.getSheet("Sheet1");
System.out.println(sheet.getRow(1).getCell(0));
System.out.println(sheet.getRow(1).getCell(1));
}

}
```

Program 2 (Contd.)

```
public class Loginwithoutfindby {
public static WebElement txtUsername;
public static WebElement txtPassword;
public static WebElement btnLogin;
@Test
public void loginapp()
{
System.setProperty("webdriver.chrome.driver","E:\\Testing\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
driver.get("https://opensource-demo.orangehrmlive.com/index.php/validatecredentials");
PageFactory.initElements(driver,Loginwithoutfindby.class);
txtUsername.sendKeys("Admin");
txtPassword.sendKeys("admin123");
btnLogin.click();}}
```

THANK YOU

SmartCliff
Career Mobility Solutions

Here is a simple example of a Runner File

```
package runner;

import org.junit.runner.RunWith;

import cucumber.api.CucumberOptions;
import cucumber.api.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
features="Features",
glue="com.Multi",
plugin={"html:target/cucumber-html-report", "json:target/cucumber.json",

"pretty:target/cucumber-pretty.txt","usage:target/cucumber-usage.json", "junit:target/cucumber-
results.xml"},
dryRun = false,
monochrome = true,
tags={"@Smoke,@Regression"}
)

public class Runner {

}
```