➢ Importance of Automation Testing: -

-Companies not only want to test software adequately, but also as quickly and thoroughly as possible. To accomplish this goal, organizations are turning to automated testing.

-To increase the test coverage

-Reduces the need for manual testing and discovers defects manual testing cannot expose and also manual testing is error prone and a time-consuming process.

-Running the tests again and again gives us the confidence that the new work we added to the system did not break the code that used to work and also to make sure that the changes we introduced are working.

-Executing the tests (particularly acceptance tests) can also help us understand what portion of the desired functionality has been implemented.

➢ Why Automation Testing is required?

-Automated software testing is the best way to increase the effectiveness, efficiency and coverage of your software testing. Every software development group tests its products, yet delivered software always has defects.

-An automated software testing tool is able to playback pre-recorded and predefined actions compare the results to the expected behavior and report the success or failure of these manual tests to a test engineer.

-Once automated tests are created, they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this, savvy managers have found that automated software testing is an essential component of successful development projects.

➢ What is Selenium?

-Selenium is a robust test automation suite that is used for automating web-based application. It supports multiple browsers, programing language and platforms.

Advantages of Selenium

1. Selenium is pure open source, freeware, and portable tool.

2. Selenium supports variety of languages that include Java, Perl, Python, C#, Ruby, Groovy, Java Script, and VB Script. etc.

3. Selenium supports many operating systems like Windows, Macintosh, Linux, Unix etc.

4. Selenium supports many browsers like Internet explorer, Chrome, Firefox, Opera, Safari etc.

5. Selenium can be integrated with ANT or Maven kind of framework for source code compilation.

6. Selenium can be integrated with TestNG testing framework for testing our applications and generating reports.

7. Selenium can be integrated with Jenkins or Hudson for continuous integration.

8. Selenium can be integrated with other open source tools for supporting other features.

9. Selenium can be used for Android, IPhone, Blackberry etc. based application testing.

10. Selenium supports very less CPU and RAM consumption for script execution.

Disadvantages of Selenium

1. Selenium needs very much expertise resources. The resource should also be very well versed in framework architecture.

2. Selenium only supports web-based application and does not support windows-based application.

3. It is difficult to test Image based application.

4. Selenium need outside support for report generation activity like dependence on TestNG or Jenkins.

5. Selenium does not support built in add-ins support.

6. Selenium user lacks online support for the problems they face.

7. Selenium does not provide any built in IDE for script generation and it need other IDE like Eclipse for writing scripts.

8. Selenium script creation time is bit high.

9. Selenium does not support file upload facility.

10. Selenium partially supports for Dialog boxes.

➢ Locators for Selenium: -

-Locators are the way to identify an HTML element on a web page, and almost all UI automation tools provide the capability to use locators for the identification of HTML elements on a web page.

-Locators are one of the essential components of Selenium infrastructure, which help Selenium scripts in **uniquely identifying the WebElements** (*such as text box, button, etc*.) present of the web page.

-DOM can be accessed in Google Chrome either by pressing ***F12*** or by ***right click*** on the web page and then by selecting ***Inspect.***

-Once we click on the "**Inspect option**", it will open the Developer Tools console. By default, it will open the "**Elements**" tab, which represents the complete DOM structure of the web page. Now, if we hover the mouse pointer over the HTML tags in the DOM, it will highlight the corresponding elements it represents on the webpage.

Selenium supports the following locators:

1)ClassName – A ClassName operator uses a class attribute to identify an object.

2)cssSelector – CSS is used to create style rules for webpages and can be used to identify any web element.

3)Id – Similar to class, we can also identify elements by using the 'id' attribute.

4)linkText – Text used in hyperlinks can also locate element

5)name – Name attribute can also identify an element

6)partialLinkText – Part of the text in the link can also identify an element

7)tagName – We can also use a tag to locate elements

8)xpath – Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.

Note: -

1) Do not use ***dynamic attribute*** values to locate elements, as they may change frequently and result in breakage of locator script. It also severely affects the maintainability, reliability, and efficiency of the automation script.

2) ***ID*** and ***name*** attributes take precedence over other locators if web page contains unique ID and name, then it's always advisable to use them instead of XPath as they are faster and more efficient.

3) While using ***locators***, make sure that locator points precisely to the required element. If the needed scenario needs to perform some operation on a single element, then make sure that locator exactly matches to only one element. If the locator points to several different elements, then it may cause breakage in our script.

……

➢ Maven Project in Eclipse

-***MAVEN*** helps us in creating the project structure, managing, and downloading the dependencies. We need to define the required dependencies in pom.xml. We will look more in detail. (By default, maven adds few dependencies specific to project structures and that will be based on the archetype that we choose).

-Maven is a build / project management tool, based on the concept of a *project object model (POM)* contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc.

-Install Maven in Eclipse

-Google Search **->** Download Maven **->** Click on link (***https://maven.apache.org***)

-> Open Maven Website **->** Scroll Down**->**Files**->**Binary zip archive**->**Click On '*apache-maven-3.8.4-bin.zip*' **->** Downloading Start.

-> After Completion of downloading, Move that Zip file to the respective location.

-> Extract that downloaded file.

-> Set Path for Maven

-We need to set up is the PATH variable, the reason we do this is that we can trigger maven executable command from anywhere in your command prompt and it will point to the MAVEN executable which is in the BIN folder.

-> Open My Computer**->** Right click on the black space**->**Properties**->**Advance System Setting**->**Environment Variable**->** Check only System variables**->**Click On New **->**Variable Name **-** "MAVEN HOME", Variable Value **-**Paste the path upto bin folder of Apache maven. **->**click Ok.

->In System Variable, click on "Path"**->** Edit->Add-> Paste the path upto bin folder of Apache maven **->** Click Ok.

-> Click on Window Search Button**->** Search "*Command Prompt*" or "*CMD*"

->Type "***java -version***" **->** Enter -for checking Java Version.

->Type "***mvn -version***" **->** Enter -for checking Maven Version.

- Create a Maven project In Eclipse

Step 1: Navigate to File Menu -> New, and check if Maven Project is displaying if yes, click on Maven Project. If NOT, please click on Other.

Step 2: After Clicking on Other, it will display 'select a wizard' screen, please type as Maven in the text box to filter and find Maven. After that Select Maven and Click on next button.

In the next screen, it will ask us to select project name and location. By default, 'Use default workspace location' will be selected. Just click on next button to proceed without making any changes.

Step 3: In the next screen, It will ask us to select an **[** ***Archetype maven-archetype-quickstart version-1.4*****]**. Depending on the type of project that we are working, we need to choose the archetype.

Step 4: It will ask us to 'Enter a group id for the artifact.' Version number will be by default 0.0.1-SNAPSHOT

Step 5: Click on Finish. Now Creating a maven project is Done.

After that, it has downloaded Maven Dependencies. We can see by expanding the Maven Dependencies. And also, it has created pom.xml file.

- Dependency for WebdriverManager In POM.XML File: -

-WebDriverManager in Selenium, is a class that allows us to download and set the browser driver binaries without us, as developers, having to put them in automation scripts manually.

-Automates the management of WebDriver binaries.

-Downloads the appropriate driver binaries, if not already present, into the local cache.

-Downloads the latest version of the browser binary, unless otherwise specified.

-Eliminates the need to store driver binaries locally. We also need not maintain various versions of the binary driver files for different browsers.

-In Maven project, we need to add the following dependency in Pom.XML -

->Google search->" WedriverManager Maven Dependency" -Enter

-> Go to on Website- "*https://mvnrepository.com* " -> Click On updated Version of WebDriver Manger-> Copy Webdriver Manger Dependency as below and paste in Pom.XML file.

```
<dependency>
   <groupId>io.github.bonigarcia</groupId>
   <artifactId>webdrivermanager</artifactId>
   <version>5.0.3</version>
</dependency>
```

➢ What is pom.xml?

-POM is an acronym for Project Object Model. It contains information about project and configuration of the project such as dependencies, build directory, source directory, test source directory, plugins, goal, etc. Maven reads pom.xml and then executes goal.

➢ Why should we use Maven?

1) We need maven to get all our dependencies automatically, which also allows users to reuse same jars across multiple projects.

2) Each and every engineers in a project use the same jar dependencies due to the centralized POM.

3) It provides the complete structure with naming conventions which is easy to locate and execute tests.

4) We can automate the complete build procedure etc.

➢ Maven Repository: -

-Maven repository is a directory where all the dependencies such as jars, library files, plugins, or other artifacts that will be required by the projects are stored.

-These repositories help us to store and maintain useful resources so that they can be used in our maven projects while building and deploying the artifacts.

-All the layout and structure of the underlying repositories of maven of any type are completely hidden for maven users.

-Type of Repositories: -

1) Local Maven Repository

2) Remote Maven Repository

3) Central Maven Repository

1) Local Maven Repository: -

-Local repositories are the folders or directory on our machine where maven is used i.e., a local machine that contains the cached jars and dependencies required by your project and other artifacts that are built by us on our machine but not released yet.

-The operations that the maven user can perform is to clean the local repository if any memory or space-related issue is there on their local repository or completely erase the data on this repository that will further lead to redownloading of the dependencies required by our project.

2) Remote Maven Repository

-The remote repositories are the resources that are located remotely and contain directory having jars, library files, and plugins, and other artifacts that might be useful and are accessed by using the protocols such as https:// of file:// protocol.

-These remote repositories are further segregated into the repositories that are kept at a remote place and shared between multiple individuals and developers for the private use of a company or the third party to maintain and share artifacts.

-We can mention our remote repositories inside the settings.xml file in the repositories tag and can add multiple repository tags for multiple remote repositories.

3) Central Maven Repository

-The second remote repository in the maven is the central repository that is available for downloading the dependencies in our maven project and also to release the artifacts that we want to on the central repository.

-The link for the central maven repository is *https://repo.maven.apache.org/maven2* while the site that is available for searching the maven dependencies available in the central repository of maven is – https://search.maven.org/.

-This type of remote repository is also called and referred to as the central repository and is available to all and is open source.

-This repository is provided by the maven community. There is no need for any configurations in settings.xml file for a central repository.

➢ Selenium WebDriver: -

- Selenium WebDriver was the first cross-platform testing framework that could configure and control the browsers on the OS level. It served as a programming interface to create and run test cases.
-WebDriver performs actions on web elements. It supports various programming languages like Java, C#, PHP, Python, among others. It can also be integrated with frameworks like TestNG and JUnit for test management.
-Selenium WebDriver supports multiple web browsers to test and run applications on.

-We have to download separate drivers and we have to specify the path as well with the location of the driver.

-WebDriver is an **interface** (interface have only function prototype). We cannot create Object of an Interface. If we create object of WebDriver, we will get compile time error ("*Cannot instantiate the type WebDriver*").

1) ChromeDriver-

-The most common web browser used today is Google Chrome. Google Chrome is widely used globally, making it essential for all the websites on Chrome to be tested.

ChromeDriver driver = new ChromeDriver ();

-We are creating instance of ChromeDriver. As per the java concept if We create an object using *New* keyword it will initiate constructor of that class. We have object of ChromeDriver class.

-In this case our driver object will access all the methods implemented in ChromeDriver class. Also, it has access to all the methods of WebDriver and SearchContexts interfaces which is implemented in RemoteWebDriver as ChromeDriver class is extending.

WebDriver driver = new ChromeDriver ();

-We are creating an instance of the WebDriver interface and casting it to ChromeDriver.

2) FireFoxDriver-

FireFoxDriver driver = new FireFoxDriver ();

-We are creating instance of FireFoxDriver. As per the java concept if We create an object using *New* keyword it will initiate constructor of that class. We have object of FireFoxDriver class.

-In this case our driver object will access all the methods implemented in FireFoxDriver class. Also, it has access to all the methods of WebDriver and SearchContexts interfaces which is implemented in RemoteWebDriver as FireFoxDriver class is extending.

WebDriver driver = new FireFoxDriver ();

-We are creating an instance of the WebDriver interface and casting it to FireFoxDriver.

➢ System.getProperty() And System.setProperty()

1) System.getProperty()-

- The method is used to obtain the system property. This system property is specified by the key which is the parameter for the method. To obtain the current working directory, the key used is "*user.dir*".

Program-

```
String path=System.getProperty("user.dir");
        System.out.println("Driver Path:" +path);
```

OutPut-

Driver Path: E:\ Acceleration\Software\Maven\Path_Maven Project

2) System.setProperty()-

-The setProperty method enables QAs to set the properties for the desired browser to be used in test automation.

-The setProperty method has two attributes – "propertyName" and "value." The propertyName represents the name of the browser-specific driver, and the value points to the path of that browser driver.

```
System.setProperty("webdriver.chrome.driver", path+"\\drivers\\chromedriver.exe\\");
```

.....

➢ WebDriver Methods In Selenium: -

1) driver.get (): -

-This method opens the specified URL in the current browser window. URL must be in the form of https://www.google.com. If the HTTPS is not included, then it will throw a exception "***InvalidArgumentException***". The URL to load and it should be a fully qualified URL

2) driver. getCurrentUrl (): -

- The getCurrentURL () method takes nothing as a parameter and returns URL of the web page currently loaded in the browser.

3) driver. getTitle (): -

- The getTitle () method takes nothing as a parameter and returns the page title of the currently loaded web page. If the web page has no title, it will return a null String.

4) driver. findElements (): -

-Example: driver. FindElements (By. Xpath ("//");
-Find all elements within the current page using this method.

5) driver. findElement (): -

-Example: driver. FindElement (By. Xpath ("//");
-Find the WebElement using the given method.

6) driver. getPageSource (): -

-Get the source code of the currently loaded page.

7) driver. Close (): -

-Close the current window, if there are multiple windows, it will close the current window which is active and quits the browser if it's the last window opened currently.

8) driver. Quit (): -

-Quits this driver instance, closing every associated window which is opened.

9) driver. getWindowHandle (): -

- This is used to get the address of the current browser window where it's focused on and returns the data type of String.

-10) driver. getWindowHandles (): -

- This is used to get the address of all the open browsers window and returns the data type of Set<String>.

11) driver. navigate (): -

- An abstraction allowing the driver to access the browser's history and to navigate to a given URL.
- A ***org. openqa. selenium. WebDriver.Navigation*** that allows the selection of what to do next

11) driver. manage (): -

-Gets the Option interface

……

➢ WebElement in Selenium: -

-It Represents an *HTML* element. Generally, all interesting operations to do with interacting with a page will be performed through this interface.

-WebElement is an Interface.

-Syntax

- FindElement command syntax:

WebElement eleName = driver. findElement (By.____ "));

- FindElements command syntax:

List<WebElement> eleName = driver. findElements (By.____ "));

➢ WebElement Methods in Selenium: -

1) clear()

-If this element is a text entry element, this will clear the value.

2) click ()

-Click this element.

3) findElement (By_)

-Find the first WebElement using the given method.

4) findElements (By_)

-Find all elements within the current context using the given mechanism.

5)getAttribute (java.lang.String name)

-Get the value of a the given attribute of the element. Will return the current value, even if this has been modified after the page has been loaded.

6) getCssValue (java.lang.String propertyName)

-Get the value of a given CSS property.

7) getSize ()

-What is the width and height of the rendered element?

8) getText ()

- Get the visible (i.e. not hidden by CSS) innerText of this element, including sub-elements, without any leading or trailing whitespace. Ruturn the innerText of this element.

9) isDisplayed ()

-Is this element displayed or not? This method avoids the problem of having to parse an element's "style" attribute.

10) isEnabled ()

-Is the element currently enabled or not? This will generally return true for everything but disabled input elements.

11) isSelected ()

-Determine whether this element is selected or not.

12) sendKeys (java. lang. CharSequence... keysToSend)

-Use this method to simulate typing into an element, which may set its value.

13) submit ()

-If this current element is a form, or an element within a form, then this will be submitted to the remote server.

.....

➢ Difference Between findElement and findElements: -

1) findElement () is used to find first web element on current page by provided locator mechanism such as id, xpath, css etc whereas findElements () is used to find all web elements on current page by provided locator mechanism such as id, xpath, css etc.

2) findElement () will return only first web element even provided locator locates more than one web element. findElements () returns all matching web elements. findElement () method internally calls findElements () method itself and return first indexed web element.

3) Return type of findElement () method is a ***WebElement*** whereas return type of findElements () method is a ***List<WebElement>.***

4) findElement () will throw ***NoSuchElementException*** if no matching element is found. findElements () will return ***an empty list*** if no matching element is found. This is the reason findElements () is a better way for checking non-presence of web element.

➢ Wait Methods in Selenium: -

- The wait commands are essential when it comes to executing Selenium tests. They help to observe and troubleshoot issues that may occur due to variation in time lag.

- while running automation tests, we see ***'ElementNotVisibleException***' This appears when a particular web element with which WebDriver has to interact, is delayed in its loading. To prevent this Exception, Selenium Wait Commands must be used.

- In automation testing, wait commands direct test execution to pause for a certain length of time before moving onto the next step. This enables WebDriver to check if one or more web elements are present/visible/enriched/clickable, etc.

- Implicit Wait

-Implicit wait specifies the amount of time the driver should wait when searching for an element if it is not immediately present.

-WebElement takes some time to be present or desired element is not present immediately. In this case we can use Implicit wait method.

- If we set implicit wait as 10 seconds and element is found in 5 seconds, driver will not wait for remaining 5 seconds.

- Once We set implicit wait, it will be applicable to all calls of findElement () and findElements () methods. There is no need to write implicit wait code again and again before locating an element.

- If we write implicit wait statement multiple times, time specified in latest implicit wait statement will be applicable.

For example: We specify implicit wait as 30 seconds at line no 5 and another implicit wait as 60 seconds at line no 10, then implicit wait as 30 seconds will be applicable for all calls to findElement () and findElements () methods from line no 6-9 and implicit wait as 60 seconds will be applicable from line no 11 onward.

-Syntax-

driver. manage (). timeouts (). implicitlyWait (Duration.ofSeconds(30));

- Explicit Wait

-Selenium WebDriver provides a special kind of dynamic and customization wait called Explicit wait.

-Explicit Waits also known as Dynamic Waits because it is highly specific conditioned. It is implemented by WebDriverWait class.

-Explicit wait can be used to wait till our desired condition or conditions are met or it run out of maximum defined wait time.

-Write for particular WebElement for particular condition.

Syntax-

WebDriverWait wt=new WebDriverWait (driver, Duration.ofSeconds(5));

wt. until (ExpectedConditions.elementToBeClickable(driver. findElement (By.id ("_")))). Click ();

-Thread.sleep () method: -

1)Thread is a class in JAVA. sleep () is not provided by Selenium WebDriver, so it is not a selenium wait.

2)sleep () is a static method of Thread class so we can use it using class name i.e., Thread.

3)Thread.sleep causes the current thread to suspend execution for a specified period.

4)sleep () methods accept duration in milliseconds. (1 s= 1000 ms).

5)It throws IllegalArgumentException – if the value of millis is negative.

sleep () throws a checked exception which we must either throw or handle it using try catch.

6)Syntax: ***Thread.sleep(3000)***;

7)It is a static wait. Execution will be on hold till specified time.

8) Disadvantage-We make WebDriver to wait for 5 seconds. What if search result appears in just a second? WebDriver will still wait for another 4 seconds. It will increase test execution time.

-FluentWait

-FluentWait is class in selenium which implements ***Wait*** interface.

-*WebDriverWait* is **subclass** of *FluentWait*.

-It is an implementation of the Wait interface that may have its timeout and polling interval configured on the fly.

-Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as ***NoSuchElementExceptions*** when searching for an element on the page.

-**Where we should use FluentWait?**

-When we do not find suitable expected wait condition in explicit wait.

-To handle dynamic web elements.

-We need to do more then just waiting.

-When we need to create your own customized wait conditions.

-About NoSuchElementException?

1) NoSuchElementException is a sub class of Runtime Exception.

2)It is a unchecked exception.

3)It is thrown by findElement (By by) method above. Note only findElement () method not findElements () method as it gives us an empty array rather than exception.

4)It is thrown when WebDriver is not able to locate an element for some reasons. It might be wrong locator or element is not loaded yet.

.....

➢ Select (*Single Select / MultiSelect in Dropdown*): -

- What is <select> tag?

- The <*select*> element, used along with one or more <*option*> elements, creates a drop-down list of options for a web form. The <*select*> element creates the list and each <*option*> element is displayed as an available option in the list.

-Drop-down can be of two types:

1. Single select drop-down*: We can select only single value at a time.*

2. Multi select drop-down: *We can select more than one value at a time.*

-Handling <select> tag drop-down in Selenium:

-Selenium developers provide a special class called "*Select*" to handle drop-down.

- "*Select*" class is present in "*org. openqa. selenium. support.ui*" package.

- "Select" class models a SELECT tag and provides helper methods to select and deselect options.

- "*Select*" class can be used only for SELECT tag drop-down.

-What happens when we try to use "Select" class for non-SELECT tag dropdowns.
"*Select*" class provides an argument constructor which accepts a web element which should be a <select> tag web element. This constructor checks that the given web element is a SELECT tag or not. If it is not, then an *UnexpectedTagNameException* is thrown

-How to check if any drop-down is Single select or Multi select:

-*Select* class provides a method named isMultiple () which returns a boolean.

-If drop-down is multi select, isMultiple () will return true otherwise false.

Program: -

```
// To check if drop-down is multi select

boolean status= selectObject.isMultiple ();
if(status==true) {

 System.out.println("Drop-down is multi select.");

} else {

 System.out.println("Drop-down is single select");

}
```

Output: -

```
Drop-down is multi select
```

-How to print all options available in drop-down:

-Printing all options available in a drop-down is same for single select and multi select drop-down. We need to use ***getOptions ()*** method of Select class.

-Select class provides a method named ***getOptions ()*** to get all available options in a drop-down.

-Return type of ***getOptions ()*** method is a List<WebElement>. Reason of returning List is a drop-down may have duplicate values as well. So, to retrieve all options, selenium developers use List as a return type of getOptions () method.

-After getting List<WebElement>, we need to iterate elements of List and use getText () method to get option value.

-How to select values from multi select drop-down:

- Method used for selecting values from multi select drop-down are same as for single select drop-down.

-In single selected drop-down, only one value will be selected but in multi select drop-down, whenever you will call selectBy___ (), new options will be selected and previously will not be affected.

- We can select a value from drop-down using three ways:

1) selectByVisibleText ();

2) selectByIndex ();

3) selectByValue ();

-How to get all selected options in a multi drop-down:

-Select class provides a method named *getAllSelectedOptions* () which returns a List<WebElement>.

-We need to call *getText* () method of web elements of List to get selected options

-How to deselect all values from drop-down:

-Select class provides a method named *deselect All* () which will deselect all selected options.

-Note-, This method is only for multi select dropdown. We cannot use with single select drop-down. If We use with single select drop-down, we will get exception *java.lang.UnsupportedOperationException.*

-How to deselect options one by one:

- Select class provides three methods to deselect option.

1) deselectByVisibleText ();

2) deselectByIndex ();

3) deselectByValue ();

- Above methods are used only for multi select dropdown. You we not use with single select drop-down. If you use with single select drop-down, you will get exception *java. lang. UnsupportedOperationException.*

-Click on drop down and iterate through options: -

-Click on drop down to make options visible.

-Get all options of drop down.

-Iterate through it and match with option to be selected.

-If found then click on option for selection.

-Optimal logic to select value in drop down without iteration: -

-Click on drop down to make options visible.

-Create a custom locator using option to be selected.

-Locate it and select.

➢ Window Handling In Selenium: -

-A window handle is a *unique identifier* that is assigned to each window created. Selenium WebDriver provides us two methods getWindowHandle () and getWindowHandles () which are used to get window handle/s.

- getWindowHandle ():

-It returns an opaque handle to this window that uniquely identifies it within this driver instance.

-We can switch to any window using its unique window handle.

-Window handle is alphanumeric.

-Return type of getWindowHandle () is a *String* type.

- getWindowHandles ():

-It returns a Set of window handles open by particular driver instance.

-It returns a Set because handle of each window is unique and Set allows only unique elements.

-Using this method, we can iterate over any window open by current instance.

- return type of getWindowHandles () is *Set<String>*

Syntax -Switch to other window-

driver. switchTo (). Window ();

- *switch to*: This method helps to switch between the windows

- Which exception is thrown when we provide wrong window handle?

- NoSuchWindowException

-What is the difference between the getwindowhandle () and getwindowhandles ()?

- *getwindowhandle* (): This is used to get the address of the current browser window where it's focused on and returns the data type of String.

- *getwindowhandles ()*: This is used to get the address of all the open browsers and returns the data type of Set<String>.

➢ Frame Handling In Selenium: -

- HTML frames are used to divide browser window into multiple sections where each section can load a separate HTML document. A frame can be created using <frame> tag in html.

-What is an iFrame?

-IFrame stands for inline frame which is also a HTML document.

-An IFrame is an HTML document embedded inside another HTML document on a website. The IFrame HTML element is often used to insert content from another source, such as an advertisement, into a Web page.

-An IFrame can have its own scrollbars and boarders.

-We can create an IFrame using <iframe> tag of html.

-We can switch to frame/iframe using four ways:

1) Switch to frame/iframe using its id.

2) Switch to frame/iframe using its name.

3) Switch to frame/iframe using its index. -*Not useful. If developer make changes in coding, then it will not work.*

4) Switch to frame/iframe using its web element. -***Useful Method***

Syntax

driver. switchTo (). frame(frame1);

-Switching back to parent frame or default content:

1) driver. switchTo (). defaultContent (); (*iframe to root html*)

-It selects either the first frame on the page, or the main document when a page contains iframes.

-It returns driver focused on the parent frame.

2) driver. switchTo (). parentFrame (); (*iframe to parent iframe*)

-It changes focus to the parent context. If the current context is the top-level browsing context, the context remains unchanged.

-It returns driver focused on the parent frame.

.....

➢ Browser Operations in Selenium: -

-Navigation commands are those commands in Selenium that are used to perform different operations such as backward, forward, refresh, wait, etc. in the browser's history.

-The navigate method is present in the WebDriver interface. It returns an object of type Navigation, which is nested interface of WebDriver interface.

-We can attain a Navigation object by calling navigate () method on WebDriver object.

1) driver. Navigate().to("https://www.google.com");

-This method Loads a new web page in the current browser window.

-It accepts a String parameter and returns nothing.

2) driver. Navigate (). Back ();

- It is used to navigate to the previous page in the browser history.

3) driver. Navigate ().forward ();

- It is used to navigate to the next page in the browser history.

4) driver. Navigate (). Refresh ();

- This method is used to refresh the current web page.

-It performs the same function as pressing F5 in the browser.

.....

➢ Alert Handling in Selenium: -

-Alert Popup: -

- It is a simple popup which shows some information to user in a small box within browser area.

-It will have one "OK" button and close button.

- Popup will be disappeared when user clicks on either OK button or close button.

- We cannot inspect alert popup. We cannot handle it with selenium webdriver directly. So we need to use **Alert** interface.

-Alert is an interface in selenium library. It has below methods in it:

1) accept () : To click on ***OK*** button on popup.

2) dismiss (): To click on ***close/cancel*** button of popup.

3) getText (): To get text present on popup**.**

4) sendKeys (java.lang.String keysToSend): To type on prompt popup

-Handling of alert popup in selenium webdriver:

To handle alert popup, first we need to switch to alert using **switchTo ()** method. Syntax is as below:

Alert alt= driver. switchTo().alert();

-alert () switches to the currently active popup for particular driver instance and return an Alert reference. It will throw ***NoAlertPresentException*** if no alert is present

-Confirm Popup: -

-This popup is used to ask confirmation from user to perform some action within browser.

-This popup will have two buttons "OK" and "Cancel". Both buttons will lead to different actions.

-Like alert popup, we cannot inspect it also.

-We need to use Alert interface to handle it in selenium webdriver.

-Prompt Popup: -

-This popup is used to get input from user as a text.

-This popup will have a text box with OK and Cancel button.

-Like alert popup, we cannot inspect it also.

-We need to use Alert interface to handle it in selenium webdriver.

-Alert interface provides a method to type text in text box.

-Prompt popup will have single text box only.

➢ Actions Class in Selenium: -

-What is Action in Selenium?

-Action in Selenium is an ***interface*** that provides us two methods: ***perform*** () and **build** (). These two methods or commands of action interface are implemented by the actions class.

- We Can perform mouse based and keyboard-based operations in actions class

1) build () Method: -

- The ***build ()*** method of action interface generates a composite action that contains all the actions gathered which are ready to be performed.

2) perform () Method: -

-This method is used to perform a sequence of actions without calling build () first.

-Syntax of Actions Class-

Actions act = new Actions (driver);

-Actions is an ordinary class, its object is created by using the new keyword. After creating object, we will have to pass WebDriver object reference as a parameter to its constructor. We can access all the methods provided by the Actions class.

-Mouse based Actions Methods/Commands in Selenium: -

1) click ():

- Clicks at the current mouse location.

2) doubleClick ():

-The doubleClick () method is used to perform a double-clicking at the current location of the mouse cursor.

3) doubleClick (WebElement element):

- This method is used to perform a double-clicking in the middle of the given element.

4) contextClick ():

- The contextClick () method is used to perform a **right-click** at the current mouse location.

5) contextClick (WebElement element):

- This method is an overloaded version of contextClick () method that is used to perform a ***right-click*** on the given web element. First, it moves the cursor to the location of web element and then performs right-click on the element.

6) moveToElement (WebElement):

- The moveToElement () method is used to move the mouse cursor to the center of a specified element on the web page. The web element is also scrolled into view.

7) clickAndHold ():

- The clickAndHold () method is used to left-clicking on an element and holds it without releasing the left button of the mouse (i.e. without releasing at the current mouse location).

8) release ():

- The release () method is used to release or to drop an element from the mouse. It can release the left mouse button on a web element.

9) dragAndDrop (WebElement, WebElement):

-The dragAndDrop () method is used to drag an element on to the location of another target element and drop it. We need to pass source element and target element to dragAndDrop () method of actions class where the source element will be dragged and dropped.

10) dragAndDropBy (WebElement, int, int):

-The dragAndDropBy () method is used to drag a web element on to the location of given offset and drop it.

-Keyboard based Actions Commands in Selenium: -

1) keyDown ():

-The keyDown () method is used to perform the action of pressing and holding a key. The keys can be Shift, Ctrl, and Alt keys.

2) keyUp ():

-The keyUp () method is used to release a key that has been already pressed using keyDown () method.

3. sendKeys (CharSequence):

-The sendKeys () method is used to type in alphanumeric and special character keys into the WebElement such as textarea, textbox, and so on.

-Difference between Action and Actions class in Selenium

1. Action in selenium is an interface whereas, actions is a class in selenium.

2. Action provides two methods such as build () and perform () whereas, actions class provides various methods to perform complex mouse actions and keyboard shortcuts.

3. Actions class extends object class and implements an action interface.

.....

➢ JavaScriptExecutor in Selenium:

- JavaScript helps us to interact WebElement where Selenium fails to do so.

- JavaScript ("JS" for short) is a full-fledged dynamic programming language that, when applied to an HTML document, can provide dynamic interactivity on websites."

- Sometimes we will find Selenium commands are not working properly.- JavaScript commands should be last resort if normal selenium commands are not working.

-For Example-

-Handling nested web elements. We have seen some toggle buttons, check boxes etc which are nested by some other tags like label etc. Normal selenium command "Click" will not be able to click on toggle sometimes, as it will find it is not clickable.

-Complete area of some web elements like button, checkbox etc are not clickable. We need to click on specific part of element to perform action. Selenium might fail here sometimes. In this case also "Js" is very useful.

-Scrolling is also a big problem in selenium. Using "Js", we can scroll by pixels or to the web element.

-Handling hidden elements. We can use "JS" to get text or attribute value from hidden web element which cannot be done by normal selenium methods.

-Drag and drop issues can be handled using JS.
-Selenium people provide an interface named "***JavascriptExecutor*** ", which provides declaration of methods as below:

-Syntax-

JavascriptExecutor js=(JavascriptExecutor)driver;

Object executeScript (String arg0, Object… arg1);

js. executeScript ("arguments [0]. click()",ele);

➢ Excel File Operations in Selenium:

- Apache POI (***Poor Obfuscation Implementation***) provides pure Java libraries for reading and writing files in Microsoft Office formats, such as Word, PowerPoint, and Excel. It has different components for reading and writing different Microsoft Office files:

1) POIFS for OLE 2 Documents

2) HSSF and XSSF for Excel Documents

3) HWPF and XWPF for Word Documents

4) HSLF and XSLF for PowerPoint Documents

5) HPSF for OLE 2 Document Properties

6) HDGF and XDGF for Visio Documents

7) HPBF for Publisher Documents

8) HMEF for TNEF (winmail.dat) Outlook Attachments

9) HSMF for Outlook Messages

-JDK does not provide a direct API to read or write Microsoft Excel or Word documents. We have to rely on the ***third-party*** library that is Apache POI.

-Maven Dependencies: -

-We will need to include following maven dependencies for Apache POI in our pom.xml file.

```
<! -- https://mvnrepository.com/artifact/org.apache.poi/poi -->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>5.2.0</version>
        </dependency>

<! -- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>5.2.0</version>
        </dependency>
```

-How we read an excel file in Java?

-We use **HSSF** and **XSSF** component provided by Apache POI to read and write excel in Java". or Common spreadsheet (SS) usermodel provided by Apache POI to read and write excel in Java". Common SS usermodel can read/write excel with both extensions .xls and .xlsx.

HSSF -*Horrible SpreadSheet Format.*

XSSF- *XML SpreadSheet Format.*

-We will find excel with two extensions: -

1) .xls *(Microsoft Excel 2003 file)*

2) .xlsx *(Microsoft Excel 2007 file or later)*

Apache POI provides different components for each type of extension of excel, which are given below:

.xls – HSSF (Horrible SpreadSheet Format)- We need to download "***poi***" jar files.

.xlsx –XSSF (XML SpreadSheet Format)- We need to download "***poi-ooxml***" jar files.

-If We do not want to read write excel based on its extension, we can use common spreadsheet usermodel which requires ***poi-ooxml*** and ***core poi*** libraries. Common spreadsheet is used widely now as it can read excel with any extension with same lines of code. No need of using HSSF or XSSF for workbook and sheet.

--***WrokBook/Sheet/Row/Cell***- These are the interfaces

-How to Read Excel file

Step1: - Gaining Access to Workbook

```
String FilePath = "d://filepath.xls";

FileInputStream fs = new FileInputStream (FilePath);

Workbook wb = Workbook.getWorkbook(fs);
```

Step2: - Get the access to the particular sheet

```
Sheet sh = wb. getSheet ("Sheet Name");
```

-This is to get the access to Sheet1.

```
int lastRowNum=s1.getLastRowNum();
```

-Gets the number of the last line (numbering starts from 0). Empty rows are included.

```
int physicalRowNum=s1.getPhysicalNumberOfRows();
```

- Returns the number of physically defined rows (NOT the number of rows in the sheet).

Step3: - Identify Rows.

```
Row r1=s1.getRow(0);
int lastcellNum=r1.getLastCellNum();
```

- Gets the index of the last cell contained in this row. (Numbering starts from 1).

```
int physicalcellNum=r1.getPhysicalNumberOfCells();
```

- Returns the number of physically defined cell (NOT the number of Cell in the sheet).

Step3: - Read data from cell.

```
Cell c1=r1.getCell();
String data=c1.getStringCellValue();
System.out.println("Data:"+data);
```

-The cell that contains the data, we can read the data in different formats like String, Date, Number using the different methods which are based upon the format of the cell we specify in the excel sheet.

-String – getStringCellValue ()

- Get the value of the cell as a string

- For numeric cells It throw an exception. For blank cells it return an empty string. For formulaCells that are not string Formulas, it throw an exception.

-Number - getNumericCellValue ()

-Get the value of the cell as a number.

- For strings it throws an exception. For blank cells it returns a 0. For formulas or error cells it returns the precalculated value;

-Date – getDateCellValue ()

- Get the value of the cell as a date.

- For strings it throw an exception. For blank cells it return a null.

-How to Write in Excel file

Step1: - Gaining Access to Workbook

String FilePath = "d://filepath.xls";

FileInputStream fs = new FileInputStream (FilePath);

Workbook wb = Workbook.getWorkbook(fs);

Step2: - Get the access to the particular sheet

Sheet sh = wb. getSheet ("Sheet Name");

-This is to get the access to Sheet1.

Step3: - Set a value for the cell.

s1. getRow (1). createCell (2). setCellValue ("Pass");

s1. getRow (2). createCell (2). setCellValue ("Fail");

s1. getRow (3). createCell (2). setCellValue ("Pass");

- getRow ()-> Returns the Row this cell belongs to

- createCell ()->Use this to create new cells within the row and return it.

- setCellValue ()->Set a string value for the cell.

Step4: - Write and close the file output stream

FileOutputStream fos=new FileOutputStream ("*Path of excel file*").

wb. Write(fos);

fos. Close ();

-**FileOutputStream** is an output stream for writing data to a File. It is a subclass of OutputStream, which accepts output bytes and sends them.

-**Write**-Write out this workbook to an OutputStream.

-Close-Closes this file output stream and releases any system resources associated with this stream. This file output stream may no longer be used for writing bytes.

➢ Properties File in Selenium: -

-A file with extension. properties in java, is called a *property file* which stores the data in form of **key value pair** (Like Map). We can store our locators in a properties file, and we can read it from properties file and we can use to create object repository.

- Properties' files are mainly used in Java programs to maintain project configuration data, database config or project settings etc.

-How to read data from properties file?

Step 1: Create a file with extension. properties as "Config. properties".

create properties file**->**

Navigate to File menu **->** New **->** Click on File

->The only thing we need to do is, give the file name and give the extension as

'.properties'. (Example: dataFile.properties)

-The below is the sample properties file that was used for tests. White space between the property name and property value is always ignored

browserName=CH

url=http://localhost/login.do

UserName=admin

password=manager

Step 2: Load the properties file using load () method of Properties class.

FileInputStream fis=new FileInputStream ("Path of File");

-FileInputStream ->Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data

-If File is not present in default project root location, we will get an exception like, *java.io. FileNotFoundException,* We have to provide a fully qualified path of the file as well.

Properties prop=new Properties ();

The ***java. util. Properties*** class is a class which represents a persistent set of properties. The Properties can be saved to a stream or loaded from a stream.

- Each key and its corresponding value in the property list is a string.

prop. **load**(fis); //Load method used for linking file with properties

-The program uses **load** () method to retrieve the list. When the program executes, it first tries to load the list from a file. If this file exists, the list is loaded else *IO exception* is thrown.

Step 3: Use get Property (String key) to read value using Key.

String Value=prop.getProperty(key);

System.out.println(key +" : "+Value);

- Searches for the property with the specified key in this property list.

-If the key is not found in this property list, the default property list, and its defaults, recursively, are then checked. The method returns null if the property is not found.

.....

➢ AutoIt Tool in Selenium: -

-AutoIT is a third-party tool.

- With the help of AutoIt tool (open-source tool), we can upload and download files.

-We need to call the AutoIt script after clicking on the upload button. Immediately after clicking on Upload button, the control should be transferred to AutoIt by the below statement which takes care of uploading the file.

-AutoIt v3 is a freeware BASIC-like scripting language designed for automating the Windows GUI (*Graphical User Interfaces*) and general scripting.

-It uses a combination of simulated ***keystrokes, mouse movement and window/control*** manipulation in order to automate tasks in a way not possible or reliable with other languages (e.g., VBScript and SendKeys). AutoIt

-Steps to Download AutoIt v3: -

1) Go to AutoIt website (*https://www.autoitscript.com*) and navigate to download page. It will display the latest version. The top most download is for AutoIt, ***click*** on it.

2) It will open a pop-up box, click on 'Save File' button.

3) Once the download is complete, **double click** on the .exe file and it will ask for our agreement on their terms and condition. Click on 'I Agree' button.

4) Complete the process and click on the '**Finish'** button at the end of the installation.

- In AutoIT, we can parameterized ControlSetText method. And I passed required string to ControlSetText method as below:

```
; wait for 10 seconds for upload window

Local $winId=WinWait("[CLASS:#32770]","",10)

; set focus In FileName field

ControlFocus($winId,"","Edit1")

Sleep (1000)

; send file name to upload

ControlSetText ($winId,"","Edit1","C:\Users\Admin\Desktop\AutoIt File Upload\New Microsoft Publisher Document.pub")

Sleep (2000)

; click on open Button

ControlClick ($winId,"","Button1")
```

-Save autoIt file with extension .au3. After that it should be converted in exe format. (Tool->Click on Build- Automatically converted in exe format on location)

Program- Uploading File Through AutoIT Tool: -

```
String url="https://demoqa.com/automation-practice-form";

BaseUtilities bu=new BaseUtilities ();

WebDriver driver=bu. startupUsingWM (url, "Ch");

WebElement PopUp=driver.findElement(By. cssSelector("img[alt='adplus-
dvertising"));

PopUp.click();


WebElement Ele=driver.findElement(By.id("uploadPicture"));

bu. clickByJs(Ele, driver);

Runtime.getRuntime().exec("E:\\Sheetal-
Acceleration\\Software\\Maven\\Path_Maven
Project\\Nov_9_seleinum\\AutoIT\\chromeupload.exe");
```

➢ TestNG in Selenium: -

-TestNG is a testing framework inspired from ***J Unit*** and ***N Unit***. In TestNG-NG stands for ***Next Generation.***

-Some functionalities that make it more powerful and easier to use, such as:

1) Annotations.

2) Run our tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc...).

3) Flexible test configuration.

4) Support for data-driven testing (with @DataProvider).

5) Support for parameters.

6) Supported by a variety of tools and plug-ins (Eclipse, IDEA, Maven, etc...).

7) Default JDK functions for runtime and logging (no dependencies).

8) Dependent methods for application server testing.

-TestNG is designed to cover all categories of tests: unit, functional, end-to-end, integration, etc...

- **Testng.xml** is a file that contains the test configuration, and it allows us to organize test classes and define test suites and tests.

-Advantage of TestNG: -

-It generates the HTML reports for implementation.

-Test Cases Can be Grouped & Prioritized more easily.

-Parallel testing is possible.

-Data parameterization possible

-Installation of TestNG in Eclipse: -

Steps 1: Navigate to Help-> Eclipse Marketplace.

Steps 2: Type "TestNG" in "Find" test box and click on "Search" icon. It will search and show you TestNG.

Steps 3: If TestNG is installed, it will show you "Installed" else "Install". Click on "Install" and it will be installed.

Steps 4: Click on "Confirm" button.

Steps 5: Click on radio button of "I accept the term of the licence agreement." and click on "Finish" button.

Steps 6: You will get below warning message. Click on "Install anyways".

Steps 7: Restart Eclipse.

We are done with the installation of TestNG Plugin in Eclipse now.

- Creating a TestNG class in Eclipse: -

Steps 1: - Select the menu File**->** New**->**Other**->** TestNG or Create a plain java class.

Steps 2: -Create methods.

Steps 3: -Add annotation as "@Test" each of the methods.

Steps 4: -Run as a TestNG suite.

Note: -

-A class is called TestNG class if it has annotations provided by TestNG. TestNG provides an annotation called "***Test***" which we need to use for any method.

-When we add this "**Test**" annotation, we don't Require to include "***public static void main (String [] args)***" method in our class. Main method is starting point of execution of a java program. But when We add "***Test***" annotation with methods, JVM run those methods directly.

-Method count always depends upon ***@Test*** annotation. Not on method Count.

-Whenever, we execute TestNG Class, we get 3 types of results-

1) Java Console Result

2) TestNG Console Result

3) HTML Report For Execution

-TestNG Test Case Priority: -

-In TestNG "***Priority***" is used to schedule the test cases. When there are multiple test cases, we want to execute test cases in order. Like First we need to execute a test case "***Registration***" before ***login***.

-In order to achieve, we use need to add annotation as ***@Test(priority=??).*** The default value will be zero for priority.

-If we don't mention the priority, it will take all the test cases as "***priority=0***" and execute.

-Priority should be an integer value.

-It can be negative, zero or positive number. If we write it is decimal, we must need to cast it into integer.

-TestNG will execute test methods from ***lowest to highest*** priority. Lower priorities will be scheduled first.

-Test methods without priority will have default priority of Zero and execution sequence will be decided based on ASCII value **[A= 65, Z=90 & a=97, Z=122]**.

-TestNG ignore default priority based on ASCII if priority value is provided.

-If we pass duplicate priority to test methods. In that case, TestNG will decide priority based on ASCII value.

-We can create a TestNG class with some test methods with priority and some without priority in same class *[Sequence-> - Ve Priority-> 0 Priority-> some without priority -> +Ve Priority]*

-We cannot pass priority to methods through testing.xml.

-What is the Sequence of Execution of all TestNG annotations?

The sequence of execution of all annotations in TestNG is as follows:

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

-Please note:

1) A testng xml can have single <***suite***> tag.

2) You can have multiple <***test***> tags in a <suite> tag.

3) You can have single <***classes***> tag in a <test> tag.

4) You can have multiple <***class***> tag inside a <classes> tag.

5) You can have multiple test method (@Test annotated) names inside <methods> tag inside a <class> tag

@BeforeSuite -This annotated method will run before the execution of all the test methods in the suite.

@AfterSuite-This annotated method will run after the execution of all the test methods in the suite.

@BeforeTest- This annotated method will be executed before the execution of all the test methods of available classes belonging to that folder.

@AfterTest- This annotated method will be executed after the execution of all the test methods of available classes belonging to that folder.

@BeforeClass- This annotated method will be executed before the first method of the current class is invoked.

@AfterClass- This annotated method will be invoked after the execution of all the test methods of the current class.

@BeforeMethod- This annotated method will be executed before each test method will run.

@AfterMethod- This annotated method will run after the execution of each test method.

@BeforeGroups- This annotated method run only once for a group before the execution of all test cases belonging to that group.

@AfterGroups- This annotated method run only once for a group after the execution of all test cases belonging to that group.

-Use of Invocation Count in TestNG: -

-By using of invocation Count, we can execute a test multiple time. Invocation count determines how many times test will execute.

-The keyword **invocationCount = <int>** is required.

@Test (invocationCount = 10)

-ThreadPoolSize: -

-The threadPoolSize attribute tells TestNG to create a thread pool to run the test method via multiple threads. With thread pool, it will greatly decrease the running time of the test method.

@Test (invocationCount=3, threadPoolSize=3)

-ThreadCount: -

-It's the number of tests that are run at the same time.

-Create a Suite in TestNG: -

-Suite is a collection of test cases. In order to create a suite in TestNG, right click on a project or package. In the context menu go toTestNG and Select "***Convert to TestNG***".

-By default, the suite name will be TestNG.xml. [We can change the suite name also] and then click on finish.
-In order to execute TestNG suite, right click on suite.xml file(Name of suite file) and run as TestNG Suite.

-Enable and Disable Test Cases in TestNG: -

- If the ***enabled=true*** then the test will execute and if ***enabled=false*** then the test will not execute.

Program: -

```
public class EnableDisableEx1 {
        @Test
        public void createAccount () {
                System.out.println ("create Account");
        }
        @Test(enabled=true)
        public void updateAccount () {
                System.out.println ("update Account");
        }
        @Test(enabled=false)
        public void deleteAccount () {
                System.out.println ("delete Account");
} }
```

Output: -

```
create Account
update Account
```

-Enable and Disable Test Cases in TestNG By Using Testing.xml file: -

-Disable and enable particular tests in a class using *include* and *exclude* tags from testng.xml file.

-Groups In TestNG: -

-TestNG allows us to perform sophisticated groupings of test methods.

-Using TestNG we can execute only set of groups while excluding another set.

-This gives us the maximum flexibility in divide tests and doesn't require us to recompile anything if we want to run two different sets of tests back-to-back.

-Groups are specified in testng.xml file and can be used either under the or tag. Groups specified in the tag apply to all the tags underneath.

- We group test cases based on modules name, suite name etc. We group test cases in Smoke, Sanity, Regression suites etc and whenever required we execute those suites.

- Suppose if we need to execute test cases of module “UserAccess”, we find out test cases which belong to group “UserAccess” and run it. we can decide easily what to run and not to run.

Program: -

```
public class GroupingEx1 {
@Test (groups= {"smoke","regression","UAT"})
public void createAccount () {
}
@Test (groups= {"smoke","UAT"})
public void updateAccount () {
}
@Test (groups= {"regression","UAT"})
public void deleteAccount () {
}}
```

-XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
	<test thread-count="5" name="Test">
		<groups>
			<run>
				<include name="smoke" ></include>
				<exclude name=" regression "></exclude>
				<exclude name=" UAT"></exclude>
			</run>
		</groups>
		<classes>
			<class name="org.testNGSuitEx.GroupingEx1" />
		</classes>
	</test> <!-- Test -->
</suite> <!-- Suite -->
```

- We will execute the group "***Smoke***" which will execute the test methods which are defined with group as "***Smoke***"

-How To Execute Groups In TestNG Without Using TestNG XML:-

-When we assign group or groups to test methods in TestNG, to include or exclude groups to run we use TestNG xml file.

- We can run groups without using TestNG xml file and can include, exclude groups easily compare to updation in TestNG xml file.

-Suppose we created two packages and one class in each package. Each class contains three methods which belong to different groups.

- Now we will run these groups without creating testing. Xml file. Steps are as follows:

Step 1: Navigate to Run-> Run Configurations:

Step 2: Click on TestNG

Step 3: Select project:

Step 4: After selecting project, you will see "Groups" under Run header. Click on radio button of "groups".

Step 5: Click on "Browse" button, you will see all group names there from all packages of project:

Step 6: Select groups which you want to run and click on Ok button:

Step 7: Click on Apply – > Run

-Parameter in TestNG: -

-TestNG allows us to create parameterized methods in a TestNG class.

-A TestNG class may contain more than one @Test annotated methods and every test method may accept zero or more parameters. We can pass those parameters as a whole at "**suite**" level.

-Parameters is a great functionality as it eliminates the need of hard coded data within script. Using testng.xml we can provide data to parameters.

-Parameter's annotation is used to read parameters from testng.xml for methods in a testng class.

-Attribute names provided in Parameter's annotations must be same as parameters name provided in testng.xml.

-Attributes are mapped in order as it appears in Parameter's annotation with method arguments. For example: First attribute in Parameter's annotation is mapped to first attribute in method and so on.

-Suppose you need to login to your application with different users with different roles. You cannot create multiple methods for each user with role type. Instead of that you can create a parameterized methods which will accept user type and login as required.

-You may need to connect different databases and each database may have different credentials. This can be achieved easily using parameterized methods.

-Please note we must need to run from testng.xml file otherwise we will get injection exception again.

- Terms Parameters and parameter in TestNG.

- "*Parameters*" is an annotation which allows us to create parameterized methods in a TestNG class.

- "*parameter*" is a tag which is used to pass values to parameterized methods of a TestNG class.

-Parallel Execution in TestNG: -

- Parallel testing or parallel execution is a process of running the test case parallelly rather than one after the other.

- Running the tests in parallel reduces the overall execution time.

- Using the parallel execution in TestNG, we can allow multiple threads to run simultaneously on the test case providing independence in the execution of different components of the software.

- Parallel Execution of Classes in TestNG

- TestNG provides an ability to run test classes in parallel. By using parallel execution of classes, each class will be started and executed simultaneously in different threads.

<suite name="My suite" parallel="classes" thread-count="5">

-Parallel Execution of methods in TestNG

- TestNG will run all your test methods in separate threads. Dependent methods will also run in separate threads but they will respect the order that you specified.

<suite name="My suite" parallel="methods" thread-count="5">

-Parallel Execution of test Tag in TestNG

-TestNG will run all the methods in the same <**test**> tag in the same thread, but each <**test**> tag will be in a separate thread.

-This allows you to group all your classes that are not thread safe in the same <**test**> and guarantee they will all run in the same thread while taking advantage of TestNG using as many threads as possible to run your tests.

<suite name="My suite" parallel="tests" thread-count="5">

-Data Provider in TestNG: -

-DataProvider is an annotation to mark a method as data provider which provides data as an array of array of Objects which can be used for test or configuration method of a testng class to run on.

-We can execute a test method for multiple set of data using DataProvider which is not possible though Parameter's annotation where we pass values from testng.xml.

-If we are using DataProvider, we do not require testng xml to run it as we need to do in case of Parameters annotation.

-You need to use it as a normal annotation with any name (Optional). For E.g. *@Test*(*dataProvider="testData"*) on top of method signature.

-A DataProvider method can have a name so that it can be used to supply data to test methods by just mentioning its name.

-If we do not provide name to a DataProvider method, we can call it with method name on which it was used.

-Return type of DataProvider annotation is an array of array of objects.

➢ GitHub: -

- GitHub is a code hosting platform for version control and collaboration. We can create repositories of our project and save our project there.

- It gives other people a way to navigate our repos.

- It makes repo collaboration easy (**e.g.**, multiple people contributing to the same project).

- A ***Version Control System*** is a utility or software which helps us to track and manage changes to a file system or code repo for better collaboration among developers to develop an application. It can keep history of changes into a file including who has changed and what was changed.

-Install GIT On Windows: -

-Navigate to the download page of GIT. (https://git-scm.com. On this page, you will find the latest version of GIT and the download link for Windows. It automatically shows the download link based on the operating system of our PC.

-Click on the "*Download for windows*" link. you will see download is happening. It will download a file named Git-2.32.0.2-64-bit. Please note download file has the current GIT version.

- Double click on the downloaded executable file of GIT. It may show a popup for permission to make changes which you should allow.

-It will show an information page containing terms and conditions. If you are interested to read all those paragraphs then do that and click on Next otherwise you can directly click on the Next button.

-Select the destination location [***c:\Program Files\ Git***] where GIT will be installed. You can continue with the default location as shown below or you can change it. Click on Next.

- In this step you need to select components to be installed. I will recommend going with the default selection. ***GIT Bash*** and ***GIT GUI*** are major components that are default selected. You can also check Additional icons on the Desktop. Click on Next.

- Select if you would like to GIT shortcut in the start menu. Go with default selection and click on Next.

-Select the default editor of GIT. We use Notepad++ as the default editor used by GIT. You can select any option in the dropdown. There are many options available.

-Now this is a new feature of GIT. If you are little aware of GIT then you may know the default branch name used by GIT is ***master***. This can be customized now. As of now, you can go with the default branch name and click on Next.

-Now you need to set up how you would like to use GIT from the command line. Use the recommended option as there will be no need to set up any environment path. Click on Next.

-Select SSL/TLS library used by GIT to use for the HTTPS connection. Go with default selection and click on Next.

-Configure how GIT should treat line endings in text files. click on Next with default selection.
-Configure the terminal emulator to use with GIT bash. As we have already selected to install GIT Bash as well. Click on Next with default selection.
-Configure the default behavior of the git pull command. continue with the default selection and click on Next.
-Choose a credential helper. It has a new option now called Git Credential Manager Core. It is a cross-platform version. Go with default selection and click on Next.
-Enable file system caching to boost performance which is default selected. Click on Next.

-Configure experimental options. These two features are new and optional. suggest you check "Enable experimental built-in file system monitor ". This will be the last screen of setup and now click on Install.

- Installation will start and should take less than 1 minute.

-You will get below screen after successful installation of GIT. Check "Launch Git Bash" and Click on Finish.

-What is git init?

-The "git init" command creates a new empty git repository or initializes an existing directory as a git directory. Once the directory is a git directory you can run any git commands.

-Can we rerun the "Git init" command again on a git initialized repo?

– Yes, we can, and It will not override an existing ". git" configuration.

-What is git Status?

-***git status***" shows that there are untracked files. It displays the state of the working directory or of the branch where you are currently working. It lists which files are staged, unstaged, and untracked.

Untracked files are files that are present in the working directory but have not been added to the repo's tracking index. In short, git is not tracking any changes to those files till we explicitly add them using the ***git add*** command.

-When we run git status commands then it actually compares what has changed from the last commit of the currently checked-out branch.

-What is git add?

-When we add a new file that time it was untracked. After running the "***git add***" command file status moved to "changes to be committed ". It is also called file is staged for commit or file is in a staging area.

Git Command

A) <push>

1) git init ->Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.

2) git add->Stage all changes in <directory> for the next commit. Replace <directory> with a <file> to change a specific file.

3) git config --global user.name"------"->Define the author name to be used for all commits by the current user.

4) git config --global user. email"------"->Define the author email to be used for all commits by the current user.

5) git commit -m "------"->Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message.

6) git remote add origin 'paste url of repository'->Create a new connection to a remote repo.

git push -u origin master-> Push code to local repository to remote repo.

****

B) <pull>

-**Create folder->Right click and Select** git bash here.

1) git clone ' Repository url paste here'->Clone repo located at onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH

****

C) <Change branch>

1) git branch->List all of the branches in your repo. Add a <branch> argument to create a new branch with the name <branch>.

2) git branch 'branch name'->Create new branch.

3) git checkout 'branch name'-> Create and check out a new branch named <branch>. Drop the -b flag to checkout an existing branch.

4) git push -u origin 'branch name'=> push code old to new branch

5) git merged 'branch name'-> Merge into the current branch.

6) git branch -d <branchname>->Delete the branch.

****

C) <Other>

1) git status-> List which files are staged, unstaged, and untracked.

2) pwd-> Present Working Directory

3) cd->Change Directory

4) git stash clear-> Remove all stashed entries

5) git rm -r [file-name.txt]->Remove a Folder

6) git diff=> Show **difference** between working directory and last commit

7) git log -p-> Display the full diff of each commit.

****

pull requests ===> Move subbranch to master branch

Question->> conflict issue**->**commit**->**pull**->**commit**->**Push

.....

➢ Assertions in TestNG

-Asserts helps us to verify the conditions of the test and decide whether test has failed or passed. A test is considered successful ONLY if it is completed without throwing any exception.

-There are two types of Assertion: -

1) Hard Assertion

2) Soft Assertion

1) Hard Assertion

-Hard Assert throws an ***AssertException*** immediately when an assert statement fails, and test suite continues with next **@Tes**t because it marks a method as failed if the assert condition gets failed and the remaining statements inside the method will be aborted

***Assert.assertEqual (String actual, String expected*);**

- It takes two string arguments and checks whether both are equal, if not it will fail the test.

Assert.assertEqual(String actual, String expected, String message)

-It takes three string arguments and checks whether both are equal, if not it will fail the test and throws the message which we provide.

2) Soft Assertion

-SoftAssert don't throw an exception when an assert fails, but it records the failure. The test execution will continue with the next step after the assert statement.

-Calling ***assertAll ()*** will cause an exception to be thrown if at least one assertion failed.

SoftAssert sa=new SoftAssert ();

We should instantiate a SoftAssert object within a @Test method. Scope of SoftAssert should only be within the Test method

➢ Maven Project Using Command Line: -

-Maven contains a wide set of commands which we can execute. Maven commands are a mix of build life cycles, build phases and build goals.

1) Maven Version- [mvn -version]

-To verify whether maven is installed or not. it will display the version of maven and Jdk including the maven home and java home.

2) Maven Clean-

-Maven Clean is used to clean the target folder. This is where the previous build's files, libraries, reports, output files, etc., are saved. The command to execute this is '***mvn -clean'***.

-By default, it discovers and deletes the directories configured in *project. build. Directory*, *project. build. output Directory, project. build. testOutputDirectory*, and *project. reporting. output Directory.*

3) Maven Install-

-Maven Install is used to install all the dependencies and create the deployment file. After this, it will run the test.

4) Maven Run-

-Maven Run will simply run the test without creating any deployment file.

5) Maven Compiler-

-The Compiler is used to compile the sources of your project.

-The Compiler has two goals. Both are already bound to their proper phases within the Maven Lifecycle and are therefore, automatically executed during their respective phases.

compiler: compile- It is bound to the compile phase and is used to compile the main source files.

compiler: testCompile - It is bound to the test-compile phase and is used to compile the test source files.

6) Maven Package-

-Builds the project and packages the resulting JAR file into the target directory.

7)Maven Resources: -

-The ***Resources*** handles the copying of project resources to the output directory. There are two different kinds of resources: main resources and test resources.

-The difference is that the main resources are the resources associated to the main source code while the test resources are associated to the test source code.

-Thus, this allows the separation of resources for the main source code and its unit tests.

8)Maven Site: -

-The Site Plugin is used to generate a site for the project. The generated site also includes the project's reports that were configured in the POM.

.....

➢ Maven Build Life Cycle: -

-A ***Build Lifecycle*** is a well-defined sequence of phases, which define the order in which the goals are to be executed.

- There are three built-in build lifecycles: **default**, **clean** and **site**.

- The **default** lifecycle handles your project deployment, the **clean** lifecycle handles project cleaning, while the **site** lifecycle handles the creation of your project's web site.

-Maven Build Life Cycle Phases: -

1) validate-validate the project is correct and all necessary information is available

2) compile- compile the source code of the project

3) test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

4) package - take the compiled code and package it in its distributable format, such as a JAR.

5) verify - run any checks on results of integration tests to ensure quality criteria are met

6) install - install the package into the local repository, for use as a dependency in other projects locally

7) deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

-These lifecycle phases are executed sequentially to complete the default lifecycle.

-Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

➢ Execute Testng.xml file Using Maven

-We can execute our tests using ***maven Surefire plug-in***.

-This plug-in is used during the test phase of software build lifecycle to execute tests. We need to add the below dependencies in pom.xml file. And also, we need to add TestNG dependency to the pom.xml file.

Step 1: - Write Program in Java Class

Step 2: -Create TestNG Suit.

Step 3: -Add Maven Dependency

-We need to include **TestNG** dependency in our project configuration file Pom.xml.

```
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.5</version>
    <scope>test</scope>
</dependency>
```

- We have to include ***Maven-surefire-plugin and maven-compiler-plugin*** dependency in our project configuration file Pom.xml

-Maven-surefire-plugin: Surefire-plugin is responsible for running tests that are placed in test source directory /src/test/java.

- Maven-compiler-plugin: Compiler-plugin is used to help in compiling the code and using the particular JDK version for compilation.

Maven-surefire-plugin dependency: -

```
<! -- Below plugins to 'Run Project through Batch' file -->
<build>
    <plugins>
        <! -- Below plug-in is used to execute tests -->
        <plugin>
            <groupId>org.apache. maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.2</version>
            <configuration>
                <suiteXmlFiles>
                    <!-- TestNG suite XML files -->
<! -- <suiteXmlFile>Parallel_Class_testing.xml</suiteXmlFile> -->
                    <! --<suiteXmlFile>${suiteXmlFile}</suiteXmlFile> -->
                </suiteXmlFiles>
            </configuration>
        </plugin>
```

-Maven-Compiler-plugin dependency: -

```
<plugin>
        <groupId>org.apache. maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
            <source>1.8</source>
            <target>1.8</target>
        </configuration>
    </plugin>
</plugins>
</build>
```

Step 4: - Execute the pom.xml file using the command "**mvn test**"

[Right click on POM.xml file ***->****Run As****->*** *Click on Maven Test]*

Or

[Right click on Maven Project ***->****Run As****->*** *Click on Maven Test]*

Step 5: -

-After executing the program, the report will be generated in our project folder under ***target\surefire-reports***. We can checkout default testng html reports.

➢ Execute Testng.xml using batch file

Step1: -Open notepad

Step2: -We have to add our project location. Write the below line of code.

```
set projectpath=E:\Admin-Acceleration\Software\Maven\Path_Maven Project\Nov_9_seleinum

cd %projectpath%

mvn test -DsuiteXmlFile=Grouping_testing.xml,Parallel_Class_testing.xml
```

Step3: -Save file with extension .bat [*For example: -maven.bat*]

Step3: -If we want to run this file Double click on that file [*maven.bat*]. It will start execution in CMD window**.** We don't require to write any maven command in the CMD.

-Whatever we need to do, whatever steps need to perform we can directly add in this batch file.

- When we double click on the batch file it run our maven project and that Command prompt got close immediately. So, If We want that Command prompt should remain open after running our maven project to see some errors are there or not, what is the output then some modification we can do in the batch file. Before running ***mvn test*** *we need to write command* ***cmd /k.***

cmd /k mvn test -DsuiteXmlFile=Grouping_testing.xml, Parallel_Class_testing.xml

- The **cmd /k** tells Command Prompt to issue the command that follows, and then stay open so that we can view results or type followup commands.

……

-We can execute testng.xml file in different ways.

1.From Java Class

2. From Suite [Testing.xml]

3. Using Pom.xml- [*Right click on POM.xml file* ***->****Run As****->*** *Click on Maven Test*]

[Right click on Maven Project ***->****Run As****->*** *Click on Maven Test]*

4. Command Line

5. Batch File.

……

➢ TestNG Listeners: -

- TestNG provides the @Listeners annotation which listens to every event that occurs in a selenium code. Listeners are activated either before the test or after the test case. It is an interface that modifies the TestNG behavior.
- **For example**, when you are running a test case either through selenium and suddenly a test case fails. We need a screenshot of the test case that has been failed, to achieve such scenario, TestNG provides a mechanism, i.e., Listeners. When the test case failure occurs, then it is redirected to the new block written for the screenshot.
- Listeners are implemented by the **ITestListener** interface. An **ITestListener** interface has the following methods:

1) onTestStart (): An onTestStart () is invoked only when any test method gets started.

2) onTestSuccess (): An onTestSuccess () method is executed on the success of a test method.

3) onTestFailure (): An onTestFailure () method is invoked when test method fails.

4) onTestSkipped (): An onTestSkipped () run only when any test method has been skipped.

5) onTestFailedButWithinSuccessPercentage (): This method is invoked each time when the test method fails but within success percentage.

6) onStart (): An onStart () method is executed on the start of any test method.

7) onFinish (): An onFinish () is invoked when any test case finishes its execution.

-We can create the TestNG Listeners in two ways. First, we can use the @**Listeners** annotation within the class and second way to use the within the suite.

➢ Page Object Model: -

- POM (Page Object Model) is a design pattern that is used in selenium test automation for organizing the helper code for maintainability and reducing code duplication. A page object or a page class is basically a representation of all or some of the actions and operations which can be performed on that page of your application and also acts as an interface to a page of your application under test.

-Why we need Page Object Model?

-Easy to understand.

-Proper categorization.

-Anybody should be able to locate desired lines of code easily in project folders.

-No duplicate codes.

-There is clean separation between test code and page specific code such as locators (or their use if you're using a UI map) and layout.

-Using POM, you can achieve abstraction and encapsulation.

-If there is any change in any element of a website then we only need to make changes in one class, and not in all classes.

- Under the Page object model, we create page classes that can be reused in another project.
- Can use PageFactory in the page object model in order to initialize the web element and store elements in the cache.
- TestNG can also be integrated into the Page Object Model approach.

-What Is Pagefactory?

-***PageFactory*** is a way of implementing the "**Page Object Model**". Here, we follow the principle of separation of Page Object Repository and Test Methods. It is an inbuilt concept of Page Object Model which is very optimized.

-The annotation **@FindBy** is used in Pagefactory to identify an element while POM without Pagefactory uses the **driver. findElement ()** method to locate an element.

-Example: -

-We automate the below scenario using the Page Object Model.

Step 1: Launch the site "http://localhost/login.do".

Step 2: Enter the valid credential [Username & Password].

Step 3: Click On Login button.

Step 4: Login to the site.

Step 1: Create the java Class classes. Creating TestBase class. Here we create an object of WebDriver, maximize browser, launching URL, Creating Test cases etc.

1) Launch browser and open facebook.com

2) Enter user credentials and do sign in

3) login to the site.

Step 2: Creating classes for login page to hold element locators and their page action methods.

Step 3: Execute the Program.

Program: -

1) Test case class program: -

```
public class ActiTimeLoginTC {
        private WebDriver driver;
        @BeforeTest
        public void URLOpen () {
                String url ="http://localhost/login.do";
                BaseUtilities obj=new BaseUtilities ();
                driver=obj.startupUsingWM (url, "CH");
        }
```

```
@Test
public void actiTimeLogin () {
        LoginPage lp_obj=new LoginPage(driver);
        lp_obj.isUserNameFieldDisplayed ();
        lp_obj.isPWDFieldDisplayed ();
        lp_obj.isLoginFieldDisplayed();
        lp_obj.enterUserName ("admin");
        lp_obj.enterPassword("manager");
        lp_obj.ClickLoginBtn ();
}}
```

2) Login Page class program: -

```
public class LoginPage {
    private WebDriver driver;
    //Step1: - Private WebElement
    @FindBy(id="username")
    private WebElement uNTextField;
    @FindBy(css="input[name='pwd']")
    private WebElement pwdTextField;
    @FindBy(css="#loginButton>div")
    private WebElement LoginBtn;
    // Setp2:- Constructor
    public LoginPage (WebDriver driver) {
        PageFactory.initElements(driver, this);
        this. Driver=driver;
    }
```

```
        // Step 3:- Page Action Methods
        public boolean isUserNameFieldDisplayed() {
                return uNTextField.isDisplayed();
        }
        public boolean isPWDFieldDisplayed() {
                return uNTextField.isDisplayed();
        }
        public boolean isLoginFieldDisplayed() {
                return LoginBtn.isDisplayed();
        }
        public void enterUserName(String uName) {
                uNTextField.sendKeys(uName);
        }
        public void enterPassword(String pwd) {
                pwdTextField.sendKeys(pwd);
        }
        public void ClickLoginBtn() {
                LoginBtn.click();
        }
}
```

➢ Cucumber Framework: -

- Cucumber is a testing tool that supports Behavior Driven Development (**BDD**). Specifications, called "features", are written in structured natural language.

-Cucumber executes a feature by mapping each of its steps to a "step definition" written in the programming language supported by that implementation of Cucumber. Cucumber is implemented in many programming languages including Ruby (the original), Java and JavaScript. It is also translated into many human languages.

-A **framework** defines a set of rules or best practices which we can follow in a systematic way to achieve the desired results.

-Behavior-Driven Development

-Behaviour-Driven Development (BDD) is the software development process that Cucumber was built to support.

- Cucumber was written to support the agile methodology called **Behavior-Driven Development**.

-BDD is a way for software teams to work that closes the gap between business people and technical people by:

1) Encouraging collaboration across roles to build shared understanding of the problem to be solved

2) Working in rapid, small iterations to increase feedback and the flow of value

3) Producing system documentation that is automatically checked against the system's behaviour

-Advantages of Cucumber

-It is helpful to involve business stakeholders who can't easily read code

-Cucumber Testing tool focuses on end-user experience

-Style of writing tests allow for easier reuse of code in the tests

-Quick and easy set up and execution

-Cucumber test tool is an efficient tool for testing. It is helpful to involve business stakeholders who can't easily read code

-Installation Of Cucumber Plugin In Eclipse: -

Step 1: Navigate to Help**->** Eclipse Marketplace **->**Search "Cucumber Eclipse Plugin"**->** Click on Install. **->** Click the check box "I accept the terms of the license agreement" on the license window then click Finish.

Step 2: The next step is to create a Project. Navigate to File **->** New **->** Java Project for creating a new project.

In the next screen, it will ask us to select project name and location. By default, 'Use default workspace location' will be selected. Just click on next button to proceed without making any changes.

Step 3: In the next screen, it will ask us to select an Archetype **[*maven-archetype-quickstart version-1.*4]**. Depending on the type of project that we are working, we need to choose the archetype.

Step 4: It will ask us to 'Enter a group id for the artifact.' Version number will be by default 0.0.1-SNAPSHOT

(For Example - *group id-Cucumber Project & artifact**: -** Cucumber Project**)***

Step 5: Click on Finish. Now Creating a maven Cucumber project is Done.

After that, we need to add below dependency for Selenium and Cucumber in pom.xml.

1) Selenium-Java

2) WebdriverManager

3) Cucumber-java

4) Cucumber-Junit

5) JUnit

6) Cucumber-TestNG

7) TestNG

8) Extentreports

Step 6: Right Click On project->Maven-> Update Maven Project->Select Project->

Click "Force Update of Snapshots/Releases->Click on Ok

-Create src/test/resource directory In Cucumber Framework: -

- src/test/resource directory -for Test resources available to our test code.

-src-> The **src** directory contains all the source material for building the project. It contains a subdirectory for each type (**test and resources**).

-test-> For the unit test code and resources

-resources-> The structure which is copied to the target class path given the default resource definition

Step1: -Right Click on Project**->** New**->** Click on Source Folder

Step 2: -In the next screen, Enter Folder Name" **src/test/resources**"

Step 3: -Click on Finish

-Feature File In Cucumber: -

The **feature file** is the essential segment of cucumber tool, which is used to write steps for automation testing. The steps generally follow the application specification.

-A **feature file** is usually a common file which stores feature, scenarios, and feature description to be tested.

-Cucumber features are written in the *Gherkin language* and stored in files with extension **.feature**.

-Feature: This keyword signifies that what follows is a basic description or name of the feature being tested or documented.

-The first primary keyword in a Gherkin document must always be **Feature**, followed by a **:** and a short text that describes the feature.

-We can add free-form text underneath **Feature** to add more description.

-These **description lines** are ignored by Cucumber at runtime but are available for reporting.

- **Scenario-** This keyword represents the name or basic description of a particular scenario testing the feature.

-In Gherkin language, a scenario includes the following keywords:

Given:

-Given keyword is used to specify the preconditions for executing a specific scenario. A scenario may include more than one Given statement or there can be no Given statements for a scenario.

When:

This keyword is used to specify the action, or an event performed by the user such as clicking on a button, entering data onto textbox etc. There can be multiple when statements in a single scenario.

Then:

Then keyword is used to specify the expected outcome of an action performed by the user. Ideally, when keyword must be followed by Then keyword to understand the expected result of user actions.

And:

And keyword is used as a conjunction keyword to combine multiple statements. **For Example**, multiple Given and When statements in a scenario can be combined using the keyword 'And'.

-Example-

Feature: Login page feature

Scenario: Validate login page fields

Given User is on the login page

Then UserName field Should be displayed

And password field should be displayed

And login button should be displayed

Scenario: Validate valid login credentials

Given User is on the login page

When UserName enters login credentials

And Click on the Login button

Then Dashboard page should be displayed

-Scenario Outline: -

-A scenario outline is a way of parameterization of scenarios.

-This is ideally used when the same scenario needs to be executed for multiple sets of data, but the test steps remain the same. Scenario Outline must be followed by the keyword 'Examples', which specify the set of values for each parameter.

Example:

Scenario Outline: Upload a file

Given that a user is on upload file screen.

When the user clicks on the Browse button.

And user enters <filename> onto upload textbox.

And user clicks on the enter button.

Then verifies that file upload is successful.

Examples:

|filename|

|file1|

|file2|

Parameters within the scenario outline must be specified with the characters '<' and '>'. A list of data values for the parameter must be specified using the Pipe (|) symbol.

-Background Keyword in Feature file: -

-Use **Background** only to setup a pre-condition that the user needs to know
-Occasionally you'll find yourself repeating the same **Given** steps in all of the scenarios in a Feature.

-Since it is repeated in every scenario, this is an indication that those steps are not essential to describe the scenarios; they are incidental details. You can literally move such **Given** steps to the **background**, by grouping them under a **Background** section.

-A **Background** allows you to add some context to the scenarios that follow it. It can contain one or more **Given** steps, which are run before each scenario, but after any Before hooks.

-Execution: - Before Hook -> Background -> Scenario-> After Hook

-A **Background** is placed before the first Scenario/Example, at the same level of indentation.

-You can only have one set of **Background** steps per Feature

-Example Keyword in Feature file: -

-A Scenario Outline must contain one or more **Examples** (or **Scenarios**) section(s).

-Its steps are interpreted as a template which is never directly run. Instead, the Scenario Outline is run once for each row in the **Examples** section beneath it (not counting the first header row).

-The steps can use **<>** delimited parameters that reference headers in the **examples** table. Cucumber will replace these parameters with values from the table before it tries to match the step against a step definition.

-You can also use parameters in multiline step arguments.

-Difference Between Hooks and Background in Cucumber: -

Sr No	Before Hook	Background
1	Before hooks will be run before the first step of each scenario.	A Background is run before each scenario, but after any Before hooks
2	Use Before when you have to do some technical setup before our scenario	Use background when you provide customer-readable pre-conditions to our scenarios.
3	**Comman Usage of @Before** 1) Starting a WebDriver 2)Setting up DB Connections 3) Setting up test data 4) Setting up browser cookies 5) Navigating to certain page	**Comman Usage of "Background"** 1) Any repeating Given Steps 2) Navigate to Login Page 3) Submit UserName and Password

-Step Definition: -

-A Step Definition is a Java method with an expression that links it to one or more Gherkin steps. When Cucumber executes a Gherkin step in a scenario, it will look for a matching step definition to execute.

-Each step of the feature file must be mapped to a corresponding step definition. Tags used on the Cucumber Gherkin file must be mapped to its step definition by using the tags **@Given**, **@When** and **@Then**.

-Right click on above feature file and Select "Run As **– >** Run Configuration-> Right Click on Cucumber feature->New Configuration**->**Enter Name" Login feature" **->**Click on Run.

-It will generate step definitions of above feature file steps in console window.

- You see a line in console window as "**You can implement missing steps with the snippets below:** ". Copy everything after that line and paste into Java class.

- In the above feature, a file can be mapped to its corresponding step definition file as shown below. Please note that in order to provide a link between the feature file and step definition file, a test runner file must be created.

```
public class LoginStepDef {

        @Given ("User is on the login page")
        public void user_is_on_the_login_page () {
        }

        @Then ("UserName field Should be displayed")
        public void user_name_field_should_be_displayed () {
        }

        @Then ("password field should be displayed")
        public void password_field_should_be_displayed () {
        }

        @Then ("login button should be displayed")
        public void login_button_should_be_displayed () {
        }

        @When ("UserName enters login credentials")
        public void user_name_enters_login_credentials () {
        }

        @When ("Click on the Login buton")
        public void click_on_the_login_buton () {
        }

        @Then ("Dashboard page should be displayed")
        public void dashboard_page_should_be_displayed () {
        }

        @Then ("User Should be on login page")
        public void user_should_be_on_login_page () {
        }
}
```

-Hooks in Cucumber: -

-Hooks are blocks of code that can run at various points in the Cucumber execution cycle. They are typically used for setup and teardown of the environment before and after each scenario.

-Where a hook is defined has no impact on what scenarios or steps it is run for. If you want more fine-grained control, you can use conditional hooks.

-Hooks can be conditionally selected for execution based on the tags of the scenario. To run a particular hook only for certain scenarios, We can associate a **Before** or **After** hook.

-**Order in Hooks-**Order hooks to run in a particular sequence is easy to do.

@Before (order = int): - This runs in increment order, means value 0 would run first and 1 would be after 0.

@After (order = int): - This runs in decrements order, means apposite of @Before. Value 1 would run first and 0 would be after 1.

-What are Cucumber Annotations?

-Annotations are predefined text which contain specific meaning. It allows compiler what should be done upon execution. There are few annotations in this cucumber.

-Comments in Feature File: -

-If we do not need to execute a particular scenario at a time, then we can comment that scenario.

-In Eclipse, to comment a multi-line or use block comment first select all the line to be commented and then press Ctrl + /. Similarly, to remove comments, we need to press again Ctrl +/.

-Test Runner Class: -

-TestRunner File is used to build a Communication between Feature Files and StepDefinition Files.

-Each step of the feature file is mapped to a corresponding method on the step definition file.

features: We use Cucumber features option to define path of feature file(s).

glue: We use Cucumber glue option to define path of step definition file(s).

plugin: We use Cucumber format option to generate output or test results in different types of formats.

Eg: HTML Report, JUnit or TestNG Report, Cucumber Report and So.

monochrome: We use Cucumber monochrome option to print console output in a very readable format. monochrome must be set to true in order to achieve it.

strict: We use Cucumber strict option to check if any step if not defined in step definition file.

-If any step is not defined in step definition file then it will stop an execution of program.

dryRun: We use Cucumber dryRun option to check whether all the steps from feature files has got methods and implemented or no in Step Definition File.

Before execution of program dryRun must be set to true and we need to make sure that all steps are implemented in Step Definition File.

Once we are sure that all steps are implemented then dryRun must be set to False and we should continue with Test Execution.

tags: We use Cucumber tags option when we have more number of scenarios in a single feature file which represents different purpose [Smoke, Sanity, Regression etc] in such cases we can make use tags option.

Eg: tags= {" @ Smoke" } >> It will pick only and only those scenarios which are tagged with Smoke in feature files.

-DriverFactory Class: -

-The DriverFactory (JUnit/TestNG) class contains the static ThreadLocal field which stores the drivers created for each thread.

-If you want that a variable should be accessed by same thread by which it is created, we can use ***ThreadLocal variables***. This class makes variables thread safe. Each thread of same object will have separate copy of object variables.

-The Thread Local is a class in java which belong to java. Lang package that enables you to create variables that can only be read and written by the same thread.

ThreadLocal<WebDriver> tlDriver=new ThreadLocal<WebDriver>();

-We need to add synchronized methods.

-Extent Reports: -

-As we all know, TestNG generates html reports by default, but they are not more readable and interactive. Extent Reports are the most popular reporting used with Selenium.

-Extent Reports is a customizable HTML report developed by Anshoo Arora which can be integrated into Selenium WebDriver using JUnit and TestNG frameworks.

-Extent Reports offer several advantages when compared to the built-in reports that are generated through JUnit and TestNG such as pie chart representation, test stepwise report generation, adding screenshots etc., at every test step and a presentable user interface that can be shared with all stakeholders of the project.

-Advantages of using Extent Reports

1) Customizable HTML report with stepwise and pie chart representation.

2) Displays the time taken for test case execution within the report.

3) Each test step can be associated with a screenshot.

4) Multiple test case runs within a single suite can be tracked easily.

5) Can be easily integrated with TestNG and JUnit frameworks

-Extents Report does not require any extra plugins but we need to include standard cucumber maven dependencies and cucumber. Properties file.

-We need to add below line in the test runner class for creating extend report **"com. aventstack. extentreports. cucumber.adapter.ExtentCucumberAdapter:"**

- cucumber. publish. enabled =false

-If We don't want to publish our reports to the [**Cucumber Reports**] (https://reports.cucumber.io/) service.

-Screenshots: -

-A **Screenshot** in Selenium Webdriver is used for bug analysis. Selenium webdriver can automatically take screenshots during the execution.

-It helps to

1) Understand the end-to-end flow of a web application.

2) Analyze a bug.

3) Track test execution.

4) Analyze the behavior of the test in different browsers/environments.

5) Trace and examine the reason behind test failure.

-Test cases may fail while executing the test suite in selenium automation. When tester faces any discrepancy in the verification, i.e., when there is a difference in actual and expected values, testers tend to take screenshots as proof for failure.

-Failures may occur because of below reasons:

1) Actual and Expected values are not matching

2) When there is no element

3) When the page takes more time to load

4) When an unexpected alert comes in to focus

5) When there are Assertion issues

-We can take a screenshot of a webpage in selenium using getScreenshotAs () method, Webdriver interface doesn't provide a method to take a screenshot.

-There is an interface called **TakesScreenshot** which provides the **getScreenshotAs ()** method and which selenium uses to take a screenshot.

-We can store the captured screenshot in different formats; all the formats are present in the OutputType interface.

1) OutputType. FILE

File scn = ((TakesScreenshot)driver). getScreenshotAs (OutputType. FILE);

FileHandler.copy(scn, new File (System.getProperty("user.dir")

+"\\screenshots\\screenshotName_DateTimeStamp.png"));

2) OutputType. BYTES

byte [] sourcePath = ((TakesScreenshot)driver). getScreenshotAs (OutputType. BYTES);

3) OutputType. BASE64

String sourcePath = ((TakesScreenshot)driver). getScreenshotAs (OutputType.BASE64);

scenario. attach (sourcePath, "image/png", screenshotName);

-We can convert all the above output types into png images

-How to ignore particular feature or scenario in cucumber?

-We can ignore or skip Cucumber Tests using tags. This works both for Scenario as well as Feature. We can skip a scenario, set of scenarios or all scenarios in a feature file. We can also use this with conjunction with AND or OR.

-if We want to ignore a specific scenario we just need to add **@ignore tag**

-We can ignore Cucumber Tests using tags. **not** word is used along with tags to ignore the test.

tags = ("not @smoke")

➢ Exceptions in Selenium: -

1) ElementClickInterceptedException:-

-The Element Click command could not be properly executed as the element that is receiving the Click command was concealed in some way.

2) ElementNotInteractableException: -

-This '*ElementNotInteractableException*' Selenium exception is thrown when even though the targeted web element exists on the DOM, interactions with that element will hit another web element.

3) ElementNotSelectableException: -

-This Selenium Exception occurs when the target element is present on the DOM but it cannot be interacted with as the element is not selectable. For example, this exception will be thrown when interacting with the script element.

4) ElementNotVisibleException

-The most common type of Selenium exception, that is thrown when even though the web element is present, but it is not visible. As the element is not visible, any interaction with the element is not possible.

-This scenario is commonly encountered in Selenium test automation where relevant operation (click, read, etc.) on the web element e.g., button, label, etc. is attempted but the element is hidden from the view. Another example is elements defined in HTML that have type hidden.

5) NoSuchFrameException

-The NoSuchFrameException Selenium exception is thrown when the frame to be switched-to does not exist.

-To avoid such Selenium exceptions, it is recommended to add a sanity check in the automated browser testing code regarding the mode of switching to the frame. Check if the frame index being used is proper. An additional wait of a few milliseconds (ms) can be added to ensure that the loading of the frame is complete.

6) NoSuchWindowException: -

-This exception is thrown when the window target being switched-to does not exist. These scenarios can be taken care of by using window handles in order to get the current set of active windows. Window handles can be used to perform appropriate actions on the same.

7) SessionNotCreatedException: -

-This exception occurs when the creation of a new session is not successful.

8) InvalidArgumentException: -

-The InvalidArgumentException exception is thrown when the arguments being passed are either invalid or malformed.

-A better practice is to verify that the web page under test is loaded using appropriate wait (in ms) in the Selenium test automation code.

9) InvalidElementStateException: -

-This Selenium exception is thrown when a command cannot be completed as the element is not in a valid state or the element is not enabled to perform that action. It can be caused if an operation like clearing an element is attempted on a web element that is non-editable and non-resettable.

-To handle such an exception in Selenium test automation, it is recommended to wait for that element to be enabled before the desired action is performed on it.

10) InvalidSwitchToTargetException: -

-This Selenium exception appears if the frame or window target being switched to is invalid. It is important to verify the XPath of the target frame using 'Inspect tool' before switching to that frame.

11) JavascriptException: -

-This exception is thrown when there is an issue in executing JavaScript code.

12) NoAlertPresentException: -

-It occurs when switching to an alert that is not yet present on the screen. The ideal way to handle alerts is to check whether the alert is present, post which the desired operation on the Alert () class should be called.

13) NoSuchCookieException: -

-This Selenium exception occurs in cases when cookie matching a given pathname is not present in the associated cookies of the current browsing context's active document.

14) NoSuchElementException: -

-The NoSuchElementException is thrown when the locator being used to access the element is invalid or an attempt is made to perform action on an element which is not on the DOM. In either of the cases, the element would not be found.

-To handle this Selenium exception, you should check whether the selector is correct and if it is correct, have an additional wait to ensure the appearance of the WebElement.

15) StaleElementReferenceException: -

-This Selenium exception occurs when a reference to an element is made that is not anymore on the DOM of the page. In simple words, the element is decayed or stale.

-Some of the possible reasons for this particular Selenium exception are:

a) The element could be inside an iFrame which is refreshed.

b) The page may have been refreshed and the element to be accessed is no longer a part of the current page.

c) The element could have been removed and re-added to the screen, since the element was located.

-A possible solution to handle this exception is to use dynamic XPath to find the required element in a loop and break the loop once the element is located.

16) TimeoutException: -

-The TimeoutException occurs when the command that is currently in execution does not complete within the expected time frame.

-A possible solution to handle this exception is increasing the wait time for the command to complete execution. However, an idealistic value should be chosen for the wait time else further execution can get delayed.

17) UnableToSetCookieException: -

-This Selenium exception occurs in the cases where the Selenium WebDriver is unable to set a cookie.

-An additional wait can be added so that the cookie is loaded for the domain.

18) UnexpectedAlertPresentException: -

-It occurs when an unexpected alert appears. A common reason for the occurrence of this exception is blocking of Selenium WebDriver commands caused by the appearance of modal windows/pop-ups on the page.

19) WebDriverException: -

-This is the base WebDriver exception that occurs due to incompatibility in the binding of Selenium WebDriver and target web browser.

-To handle this exception, you should download the Selenium WebDriver library that is compatible with the corresponding web browser.

……