**Selenium WebDriver Contents**

* Introduction to Automation
* Configuration
* Basic Commands of WebDriver
  + get()
  + getCurrentUrl()
  + getTitle()
  + getPageSource()
  + close()
  + quit()
* Locators
  + Name
  + Id
  + ClassName
  + CssSelector
  + Xpath
  + LinkText
  + PartialLinkText
  + TagName
  + RelativeLocator
* Handling controls (WebElement)
  + Text Box
  + Command button
  + Links
  + Checkbox
  + Radio button
  + Drop down list
  + List Box
* Synchronization
  + Thread.Sleep
  + Implicit Wait
  + Explicit Wait
  + Fluent Wait
  + PageLoadTimeout
* Handling Alert
* Handling Multiple Browser Windows
* Handling Dynamic Menus
* JavascriptExecutor
  + Scrolling the page
  + Clicking on Control
* Handling File Upload
* Robot Class
* Handling Frame
* Handling Shadow Object
* Mouse Actions via Action Class
  + Left Click
  + Right Click
  + Double Click
  + Drag and Drop
* Screenshot

**TestNG**

* Introduction
* Configuration
* Executing Single test
* Executing multiple tests
* Annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @BeforeClass
  + @AfterClass
  + @Parameters
* Data driven testing via @DataProvider
* Assertions
* Creating reports
* Modular Framework
  + Executing Single / multiple test
  + Skipping Single / multiple test
  + Executing / skipping single / multiple classes
  + Executing / skipping groups
  + Executing package
* Keyword driven framework
  + Reading data via .properties file
* Page Object Model (POM)
  + Creating utility class
  + Creating client class
* Data Driven Framework
  + Read data from Excel file
  + Write the data to Excel file
* Introduction to Hybrid framework

**Maven**

* Introduction
* Configuration
* Adding dependencies
* Executing tests
* Extent Report

**Cucumber**

* Introduction
* TDD v/s BDD
* Configuration
* Creating feature file using keywords
  + Feature
  + Scenario
  + Given
  + Then
  + When
  + And
  + Background
  + Scenario Outline
  + Examples
* Creating glue code / step definition
* Creating runner class
* Executing script via runner class
* Tags in cucumber
* Hooks in cucumber
* Data driven testing in cucumber

Software Testing

This is a process of checking **C**orrectness, **C**ompleteness, **S**ecurity and **Q**uality of developed software application.

Manual Testing

Tester is using hand-eye-brain co-ordination

Actions

* Entering the data in text box
* Clicking on buttons
* Selecting options from drop down list, list box, radio button, check box
* Navigating from one page to another
* Mark the test case as pass or fail

Automation testing means performing all above actions via a machine.

Machine in this context is Test Automation Tool.

Every automation tool is a software.

Every tool understands programming language, you are supposed to provide the instructions via any one of the supported programming language. (Selenium – Java, C#. net, JavaScript, Python)

Process of Automation

1. Planning
2. Tool selection
   1. Technology / type of application
   2. Cost of tool
   3. Market presence
   4. Support availability
   5. HR availability
3. Creating the script
4. Creating the test data
5. Executing the script
6. Generating the report
7. Maintance

Selenium

This is a bundle / suite of tools for testing web application.

Components of Selenium

* Selenium IDE
* Selenium Grid
* Selenium RC
* Selenium WebDriver

**Selenium WebDriver**

* This is the tool for testing browser based applications (Web Sites)
* Interface in Java
* API

Create 2 Folders (On D driver or E Drive)

1. YourName\_SeleniumDemos
2. Selenium Jar File

Pre-Requisite for Selenium WebDriver

1. Windows 10
2. Minimum Java 11
3. Any one updated browser
4. Any one IDE (Editor)
   1. Eclipse
   2. Idea Intellij
5. Selenium Jar file

Download Selenium Jar File

1. Launch selenium.dev
2. Click on Downloads link
3. Download Latest stable version [4.34.0](https://github.com/SeleniumHQ/selenium/releases/download/selenium-4.34.0/selenium-server-4.34.0.jar)
4. Open your downloads folder
5. Cut / Copy this file and paste in 2nd folder (Selenium Jar Files)

Configuration of Selenium with Eclipse

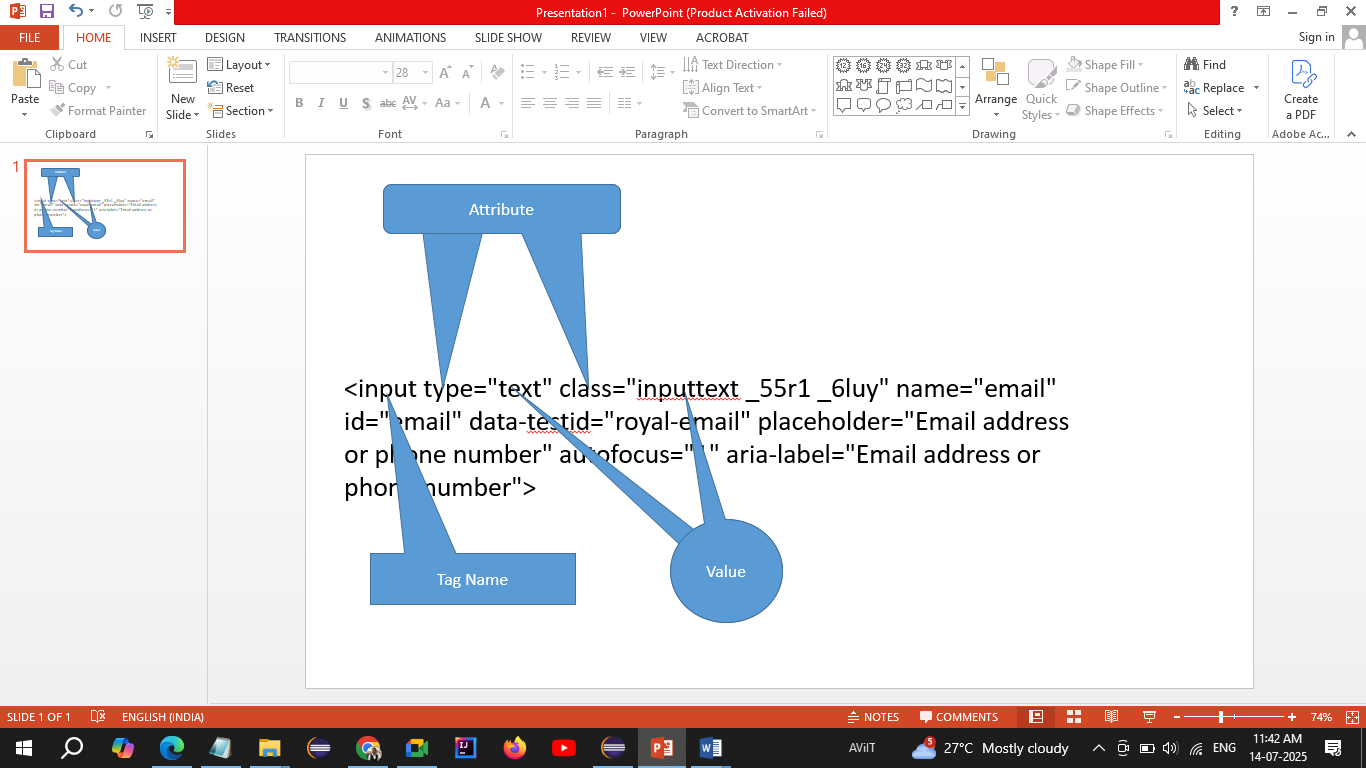
1. Launch Eclipse
2. Click on browse and
3. Select the 1st folder which you have created in earlier step
4. Click on Launch
5. Click on File menu 🡪 New 🡪 Java Project
6. Give any name to Project
7. **Uncheck create module-info.java file checkbox**
8. Click on Finish button
9. Create a package
   1. Right click on project
   2. New 🡪 Package
10. Create a Class
11. Right Click on Project 🡪 Build Path 🡪 Configure Build Path
12. Click on Libraries tab
13. Click on ClassPath
14. Click on Add External JARs…
15. Select the file which you have downloaded in earlier stpe
16. Click on Apply and Close

Methods of Selenium WebDriver

1. Launch the browser 🡪 Create object of WebDriver interface
2. get() 🡪 Launch the website via URL. Pass the absolute URL.
3. driver.manage().window().maximize() 🡪 Maximize the browser window
4. close() 🡪 Close the current browser window.
5. getTitle() 🡪 Returns the title of webpage that is opened in browser. (String)
6. getCurrentUrl() 🡪 Returns the URL of the webpage that is opened in browser (String)
7. findElement() 🡪 Locate / finds the control on the web page. It locates the first occurrence. (WebElement)
8. findElements() 🡪 Returns multiple controls on the page. (List<WebElement>)

Common Exceptions

1. InvalidArgumentException 🡪 You have given wrong URL of the site. URL should be absolute means starting with http or https
2. NoSuchElementException 🡪 Selenium is unable to locate this control due to
   1. The value of locator is wrong
   2. The value of locator is dynamic
   3. Synchronization issue
3. InvalidSelectorException 🡪
4. SessionTimeoutException 🡪 When Selenium is unable to load the page within 30 seconds.



**WebElement**

* Every control (like text box, button, radio button, dropdown list etc) is treated as WebElement in selenium
* This is an interface that represents any control on the page.

**Methods of WebElement interface**

1. sendKeys() 🡪 Enters the text in the text box
2. sendKeys(Keys.ENTER) 🡪 Hits enter from keyboard
3. click() 🡪 Click on any control.
4. getText() 🡪 Return the text on the control. (String)
5. isSelected() 🡪 Checks that whether the checkbox / radio button is selected or not. (boolean)
6. isEnabled() 🡪 Checks that whether the control is enabled or disabled (boolean)
7. isDisplayed() 🡪 Checks that whether the control is visible or not. (boolean)
8. getAttribute() 🡪 Return the value of any attribute of control. (String)

Locators in Selenium

Is the way to identify or locate any control on the web page

1. Name
2. Id
3. ClassName
4. CssSelector
5. Xpath
6. LinkText
7. PartialLinkText
8. TagName

**ClassName Locator:** If the control is having multiple classes, (Multiple classes are separated by space) then take any one of the class from multiple classes.

**CssSelector**

This is the technique via which you can read any control using any one of the **attribute.**

Types

1. Using Single Attribute  
   tageName[Attribute = “Value”]  
   input[data-testid="royal-email"]
2. Using Multiple Attributes  
   tagName[Attribute1=”value”][Attribute2=”Value”]
3. Using Special Character
   1. ^ 🡪 Starts with
   2. $ 🡪 Ends with
   3. \* 🡪 Contains

Uttur Bus Stand 🡪 Bajar Peth 🡪 Take a Left 🡪 Kumbhar Galli 🡪 House no 12 on Left Side

Xpath – XML Path

1. Absolute Xpath

Starts with html

1. Relative Xpath
   1. Taking reference of Parent tag  
      //tagName[@attribute=”value”]

**Handling Drop Down List / List Box**

* If anyone of the control is having <select> tag then only this will be treated as Drop down list / list box.
* **Select** class is used to handle dropdown list / list box

**Methods**

1. getFirstSelectedOption() 🡪 Returns the selected option / element from the dropdown list. (WebElement)
2. getOptions() 🡪 Returns list of all the elements / options from the dropdown list / list box. (List<WebElement>)
3. selectByVisibleText() 🡪 Selects the option by using the inner text on the option.
4. selectByContainsVisibleText() 🡪 Selects the option by using its partial inner text.
5. selectByValue() 🡪 Selects the option by using its **value** attribute.
6. selectByIndex() 🡪 Selects the option by using its zero based index no.
7. getAllSelectedOptions() 🡪 Returns list of all selected options / elements from the list box. (List<WebElement>)
8. isMultiple() 🡪 Checks that whether the list box / drop down list allows to select multiple options (boolean)

**Synchronization (Waits in Selenium)**

It is the process of adjusting speed of tool with speed of application.

1. Thread.sleep() 🡪 Pause the execution of the script for specified milliseconds.
   1. It takes mandatory delay.
   2. It is applicable to single statement only.
2. ImplicitWait
   1. Doesn’t take mandatory delay
   2. It is applicable throughout the script.
3. ExplicitWait (WebDriverWait class)
   1. It is applicable to single statement only
   2. It doesn’t take mandatory delay.
   3. I can check some condition like, elementToBeClickable, alertIsPresent etc.
4. FluentWait
   1. It is applicable to single statement only
   2. It doesn’t take mandatory delay.
   3. I can check some condition like, elementToBeClickable, alertIsPresent etc.
   4. Allows to handle exception as well.  
      w – withTimeout   
      i – ignoring   
      p – pollingEvery   
      u – until
5. PageLoadTimeout 🡪 You can add some time duration to avoid SessionTimeOutException.   
   By natural driver.get() waits for maximum 30 seconds to load any page, if any page is not getting loaded in 30 seconds, then Selenium will throw SessionTimeOutException.   
   With the help of PageLoadTimeout you can add more wait till the page get loaded.