**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

Pre-Requisite for Selenium WebDriver

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

**Configuration of Selenium WebDriver**

Create 2 Folders

1. YourName\_SeleniumDemos
2. Selenium Jar Files

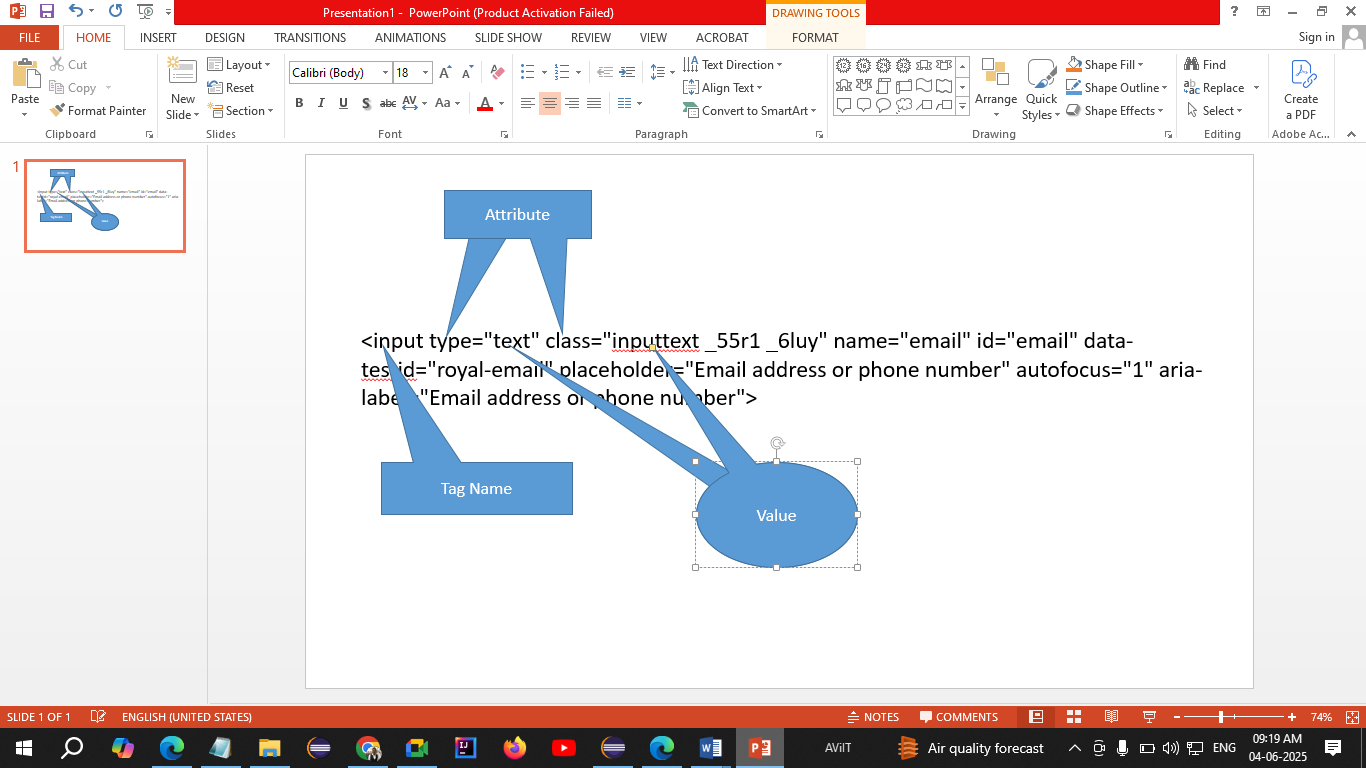
* Open Eclipse
* Create Project
* Create Package
* Create Class
* Right click on Project 🡪 Build Path 🡪 Configure Build Path
* Click on Libraries
* Click on ClassPath
* Click on Add External Jars…
* Select the jar file which we have downloaded from Selenium.dev/downloads site
* Click on Apply and Close

Methods of WebDriver

1. Launching the Browser 🡪 Create object of WebDriver interface.
2. get() 🡪 You can launch any website via URL. (Accepts absolute URL, which starts with http)
3. driver.manage().window().maximize() 🡪 Maximize the browser window
4. close() 🡪 Will close the browser window which is opened by WebDriver object.
5. getTitle() 🡪 Return the title of the page which is opened in browser. (String)
6. getCurrentUrl() 🡪 Returns the URL of the page which is opened in browser. (String)
7. getPageSource() 🡪 Returns the rendered HTML code of the page. (String)
8. findElement() 🡪 Find the single control on the page**. It always located the first occurrence**. Using locator. (WebElement)

**Common Exceptions in WebDriver**

1. InvalidArgumentException 🡪 If the URL is not in the correct format. URL should be Absolute
2. SessionNotCreated 🡪 The version of WebDriver and browser are mismatch.
3. NoSuchElementException 🡪 Selenium is not able to find the specified control. Possible reasons are
   1. The value of locator is wrong
   2. The value of locator is dynamic
4. InvalidSelectorException 🡪 The value of locator is not in the correct format.



**Locators**

Locators are the way to find any control on the web page.

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

**WebElement**

* Every control / html control / web control is treated as WebElement in WebDriver.
* It is an interface in WebDriver API.

Methods of WebElement

1. sendKeys() 🡪 Enters the text in text box. If some text is present in the text box then the new text will appended to the existing text.
2. click() 🡪 Clicks on any control.
3. getText() 🡪 Returns the text on the control. (String)

**CssSelector**

1. Single Attribute  
   tagName[attribute=”value”]  
   input[data-testid="royal-email"]
2. Multiple Attributes  
   tagName[attribute1=”value”][attribute2=”value”]
3. Special Characters
   1. ^ 🡪 Starts with
   2. $ 🡪 Ends With
   3. \* 🡪 Contains