Software Testing:

Process of checking for CORRECTNESS, COMPLETENESS, SECURITY & QUAILITY of developed software application. (CCSQ)

TC

**Objective**

To validate login functionality when valid data is provided.

**Pre-Requisite**

Login page should display & test data should ready

**Steps**

1. Enter valid user name
2. Enter valid password
3. Click on Login button

**Expected Result**

User should log in

Home page should display

**Automation Testing:**

Testing the application with the help of tools. (Automation testing Tools)

**Advantages:**

1. Time saving
2. Faster execution
3. Accuracy
4. Less manual work
5. Easy for the repetitions
6. Cost Saving
7. Improve the quality
8. Improve efficiency
9. Easy report management
10. Reduces the errors
11. Better allocation of resources
12. Less infra
13. Regression testing very easy

**When to automate:**

1. Build is stable
2. Repeated test cases
3. Huge amount of data
4. Compatibility Testing

**Tool Selection Criteria**

1. Tool support for the application
2. Availability of the tool
3. Market
4. Tester availability
5. Budget
6. Support availability for the tool
7. Reporting facility

**Process of automation**

1. Planning
2. Tool selection
3. Create the test script
4. Create the test data
5. Execute the test case (Script)
6. Report
7. Maintance

**Types of Automation Testing**

1. UNIT
2. GUI (Graphical User Interface)
3. API (Application Programming Interface)

**Selenium**

Bundle / Suite of automation testing tools.

**Components of Selenium**

1. Selenium IDE  
   Record and Playback
2. Selenium Grid  
   Parallel Execution on multiple browser and platforms
3. Selenium RC (Remote Control)  
   Create a script 🡪 Pass to RC Server 🡪 RC Server will take care of execution
4. Selenium WebDriver  
   In replacement of RC

**Selenium WebDriver**

Browser

(Chrome, FF, Edge, Safari)

Script

(Java, C#, Python)

1. Is automation testing tool for testing WEB (BROWSER) BASED APPLICATIONS
2. It is an API for communication with browser
3. It is an interface in Java.

**Pre-Requisite for Selenium WebDriver**

1. Minimum Windows 10
2. Java – Minimum 11 (java –version [on command prompt])
3. Editor (Java – IDE)
   1. Eclipse
   2. Intellij IDEA
4. Any one **updated** (latest) Browser
5. Jar File (API) (<https://www.selenium.dev/> 🡪 downloads 🡪 Download Latest Stable Version)

**Configuration:**

Create 2 folders (Better on D:)

1. **Selenium Jar File** (For storing all the jar files required for the Selenium)
2. **YourName\_SeleniumDemos** (For storing our Selenium Demos)

Open Eclipse 🡪 Select (Browse) a folder as Workspace 🡪 Launch

File 🡪 New 🡪 Java Project 🡪

Give Name 🡪 Select Java Version (Java SE 11) 🡪 Uncheck Create module-info.java file checkbox 🡪 Click on Finish

Right Click on Project 🡪 New 🡪 Package

Create a class in this package

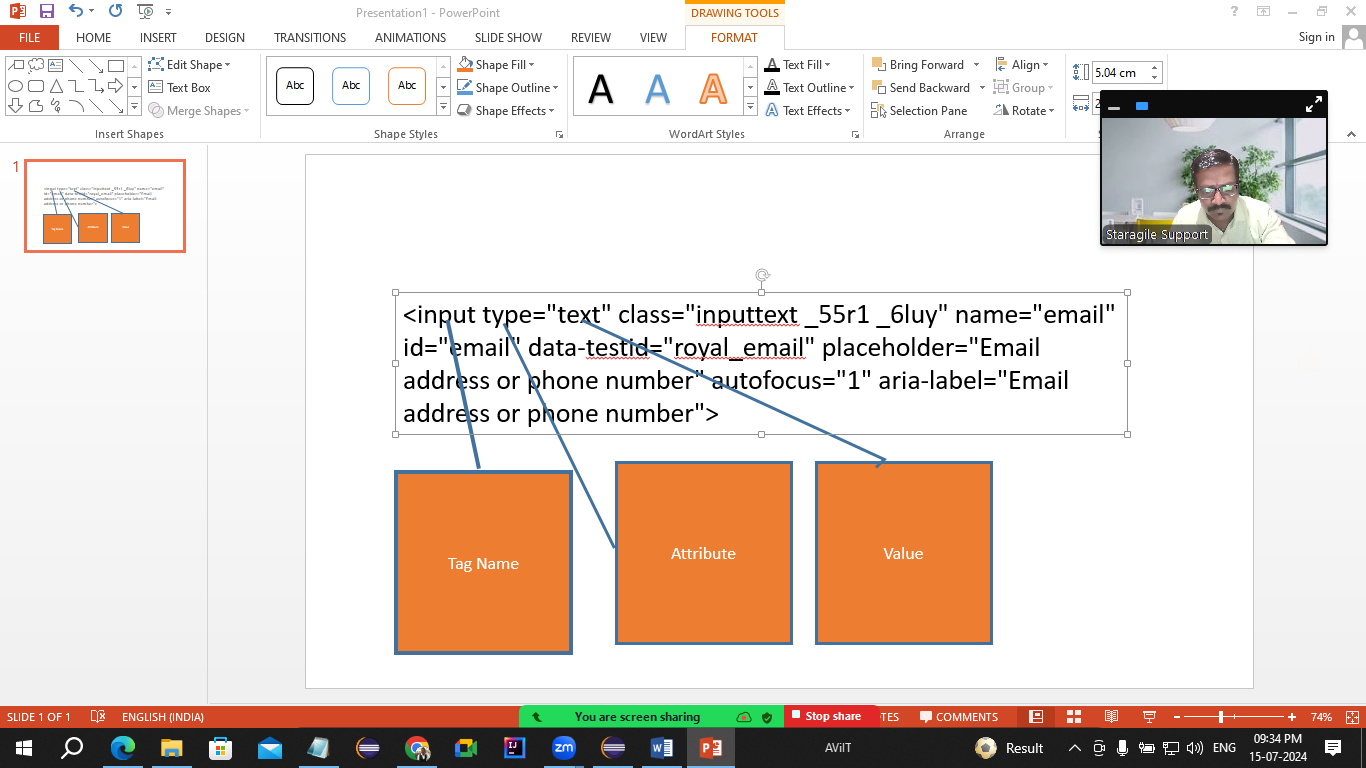
Right Click on Project 🡪 Build Path 🡪 Configure Build Path 🡪 Click on Libraries Tab 🡪 Click on ClassPath 🡪 Click on Add External JARs… 🡪 Select the Selenium Jar file which you have downloaded 🡪 Click on Open 🡪 Click on Apply and Close

**Selenium WebDriver**

1. Launch the browser 🡪 Create object of WebDriver interface
2. **get()** 🡪 Open the url
3. **driver.manage().window().maximize()** 🡪 Maximize the browser window
4. **close()** 🡪 Close the browser window which is opened by WebDriver object.
5. **getTitle()** 🡪 Return the title of the page. (String)
6. **getCurrentUrl()** 🡪 Return URL of the current page (String)
7. **getPageSource()** 🡪 Return rendered HTML (html code) of the page.
8. **findElement() 🡪** Read / Find a control (WebElement) on the page. Locates first occurrence. (WebElement)
9. **findElements() 🡪** Read / find multiple controls on the page. (List<WebElement>)

**Locator:** is the way of finding the control on the page.

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



**WebElement**

* Every control (text box, button, link, radio button, check box, dropdown list) is treated as WebElement in Selenium.
* **WebElement is an interface.**
* Which refers to the control on the page.

**Methods of WebElement**

1. **sendKeys()** 🡪 Will enter / type some text in the text box. It will append the text.
2. **click() 🡪** Will click on the control
3. **submit() 🡪** Will click on the button which is having type=”submit” attribute
4. **getText() 🡪** Will return the text on any control. (String)
5. **isSelected() 🡪** Verify that the control (Checkbox & Radio button) is selected or not. (Boolean)
6. **isEnabled() 🡪** Verify that the control is enabled or disabled. (boolean)
7. **isDisplayed() 🡪** Verify that the control is visible or not (boolean)
8. **getAttribute() 🡪** Returns the value of any attribute of the control. (String)

**Common Exceptions in WebDriver**

1. **SessionNotCreatedException** 🡪 If your selenium WebDriver and browser versions are not matching
2. **InvalidArgumentException** 🡪 The URL is not in the correct format. (URL should be absolute)
3. **NoSuchElementExceptio** 🡪 Selenium is unable to locate this control
   1. The value of locator is wrong.
   2. Locator value may be dynamic. (It is changing for every request)
   3. Synchronization issue.
4. **InvalidSelectorException** 🡪 The value of locator is not in the correct format.

**CssSelector**

(Cascading Style Sheet)

1. Using single attribute  
   tagName[attaibute = “value”  
   eg.   
   input[placeholder="Email address or phone number"]
2. Using multiple attributes  
   tagName[attribute1 = “value”][attribute2 = “value”]  
   input[placeholder="Password"][name="pass"]
3. Using special characters
   1. ^ 🡪 Starts With  
      tagName[attribute^=”value”]  
      button[id^="u\_0\_5"]
   2. $ 🡪 Ends With  
      tagName[attribute$=”value”]  
      input[data-testid$="email"]
   3. \* 🡪 Contains  
      tagName[attribute\*=”value”]

<a>Gmail</a>

**XPath**

Stands for XML Path

Pattankodoli Bus Stand 🡪 Take Right turn 🡪 Water Tank 🡪 Hupare Nagar Lane No 9 🡪 House No 1128

Types

* Absolute XPath  
  Starts with html
* Relative XPath  
  Starts with //

**Handling Drop Down List & List Box**

Select class is used to handle drop down list & list box

**Please note: if the control is having *select* tag, then only we can handle with help of Select class.**

Methods:

* getFirstSelectedOption() 🡪 Return the selected option from the drop down list. (WebElement)
* getOptions() 🡪 Returns list of all the options from drop down list. (List<WebElement>)
* selectByVisibleText() 🡪 It will select the option (Element) from the list using the text on the element. (Need to pass string value)
* selectByValue() 🡪 It will select the option (Element) from the list using the value of element. (Need to pass String value)
* selectByIndex() 🡪 It will select the option (Element) from the list using the index (zero based) of element. (Need to pass int value)
* getAllSelectedOptions() 🡪 Will return list of all selected elements from the LIST BOX. (List<WebElement>)
* isMultiple() 🡪 It will check whether we can select multiple options / whether it is list box or not. (boolean)
* deSelectByIndex()
* deSelectByValue()
* deSelectByVisibleText() 🡪 Will de select the element / option
* deSelectAll() 🡪 Will deselect all options

**Synchronization (Waits in Selenium)**

Process of adjusting speed of tool with speed of application.

Methods of Synchronization

1. Thread.sleep() 🡪 Pause the execution of script for specified number of time
   1. It is applicable to single statement only.
   2. It takes mandatory delay.
2. ImplicitWait (ImplicitlyWait)
   1. It is applicable throughout a script
   2. It doesn’t take mandatory delay.
3. ExplicitWait (WebDriverWait)
   1. It doesn’t take mandatory delay.
   2. It is applicable to single statement only.
4. FluentWait
   1. It doesn’t take mandatory delay.
   2. It is applicable to single statement only.
   3. We can manage the Exception as well  
        
      w 🡪 withTimeout  
      i 🡪 ignoring   
      p 🡪 pollingEvery  
      u 🡪 until
5. pageLoadTimeout