Module Name: Selenium WebDriver (Automation Testing)

Trainer Name: Ankush Vankore

Total Sessions: 19

Selenium WebDriver

* Introduction of Automation
* Need of Automation
* Process of Automation
* Introduction Selenium
  + Selenium IDE
  + Selenium Grid
  + Selenium RC
  + Selenium WebDriver
* Introduction to Selenium WebDriver
* Configuration
* Basic methods of WebDriver
  + Launching the different browsers
  + get()
  + getTitle()
  + getCurrentUrl()
  + getPageSource()
  + close()
  + quit()
* Locators (Identifying the controls)
  + Name
  + Id
  + ClassName
  + CssSelector
  + XPath
  + LinkText
  + PartialLinkText
  + TagName
* Handling Controls
  + Text Box
  + Command Button
  + Radio Button
  + Check box
  + Dropdown list
  + List Box
  + Links
* Synchronization
  + Thread.Sleep()
  + ImplicitWait
  + ExplicitWait
  + FluentWait
  + PageLoadTimeout
* Handling table
* Handling multiple browser windows
* Handling file upload
* Robot class
* Handling alerts
* Handling mouse actions
  + Hover
  + Click
  + Right click
  + Double click
  + Drag and drop
* JavascriptExecutor
* Handling menus
* Screenshots

TestNG

* Introduction
* Installation / Configuration
* Writing single tests
* Writing multiple tests
* Setting the priority
* Annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
* Data driven testing via @DataProvider
* Reports
  + Normal report
  + HTML Report
* Modular framework
  + Execute single test / multiple tests
  + Skipping single test / multiple tests
  + Executing / skipping groups
  + Executing single class /multiple classes
* Parameterization
* Keyword Driven framework
* Page Object Model (POM)
* Data Driven Framework via Apache POI

Maven

* Introduction
* Creating the project
* Adding dependencies
* Maven Architecture
* Extent Report
* Execution of project

Cucumber

* Introduction to BDD
* Configuration
* Components of Cucumber
  + Feature file
  + Step Definition
  + Runner Class
* Keywords
  + Feature
  + Scenario
  + Background
  + Given
  + When
  + Then
  + And
  + Examples
  + Scenario Outline
* Tags in Cucumber
* Hooks in Cucumber
* Data driven framework via Cucumber

Java Revision

* Basics of language
  + Variables
  + Data Types
  + Conditions
    - If
    - If else
    - Nested if else
    - Switch case
  + Loops
    - For Loop
    - While
    - Do – While
  + Writing and calling the function / methods
* Creating class and object
  + Constructor
  + Getter and setter methods
  + Static members and static methods
* Basics of Inheritance
  + extends keyword
  + Concrete class
  + Abstract Class
  + Interface
* Exception Handling
  + throws keyword
  + try catch block
* Collections
  + List
  + Set
  + Map

What is Software Testing?

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

**Process**

* Scenario Creation
* Test Case creation
* Test Data Creation
* RTM Creation

**Operations while testing**

Entering some data in text box

Clicking the buttons

Selecting the values from check box, radio buttons, drop down list, list box

Navigating from one page to another

Verifying the expected result

Marking the test case as pass or fail

During manual testing tester uses their hand – eye – brain co-ordination

**Automation testing** is performing above actions via a machine.

Machine in this context is **Test Automation Tool.**

**Every automation tool is a Software.**

Every tool understands some specific programming language like Java, C#, Python, Javascript, Ruby etc

**Advantages of Automation**

* Reduces the time
* Faster execution
* No human errors
* More accuracy
* Reduces the cost
* Efficiency
* Regression testing
* Huge amount of data
* 100% Test overage
* Reusability of the script
* Easy reporting

**When to Automate**

* Stable requirement
* Regression
* Data testing
* Performance testing
* Security testing
* Compability testing

**Types of Automation Testing**

1. Functional Testing
   1. Selenium WebDriver
   2. Tosca
   3. QTP
   4. Appium
2. API Testing
   1. Postman
   2. RESTFull API
3. Unit testing
   1. JUnit
   2. NUnit

**Process of Automation**

1. Planning
2. Selection of Tool
   1. Technology / Type of AUT
   2. Cost of the tool
   3. Support availability
   4. Automated reports
   5. Tester availability
3. Test Script Creation
4. Test Data Creation
5. Execution
6. Report
7. Maintance

**Selenium**

It is suite / bundle of test automation tools those are used to test **Web Based Application (Web Sites).**

**Components of Selenium**

1. Selenium IDE – Record and playback
2. Selenium Grid – Parallel Execution on multiple platform, multiple browsers
3. ~~Selenium RC – Remote Control~~
4. Selenium WebDriver

**Selenium WebDriver**

* Test automation tool to test Web Based application (Web Sites)
* It an API for automation testing
* It is an interface in Java

**Create 2 folders**

1. YourName\_Selenium Demos
2. Selenium Jar File

Pre-Requisite for Selenium WebDriver

* Windows 10 (Minimum)
* Any one updated browser
* Java 11 (JDK 11) Minimum
* Java Editor
  + Eclipse
  + Idea Intellj
* **Selenium Jar file**
  + Open selenium.dev website in any browser
  + Click on Downloads link
  + Download Latest stable version [4.34.0](https://github.com/SeleniumHQ/selenium/releases/download/selenium-4.34.0/selenium-server-4.34.0.jar)
  + Open your downloads folder
  + Copy this file and paste in the 2nd folder (Selenium Jar Files)

**Eclipse Configuration**

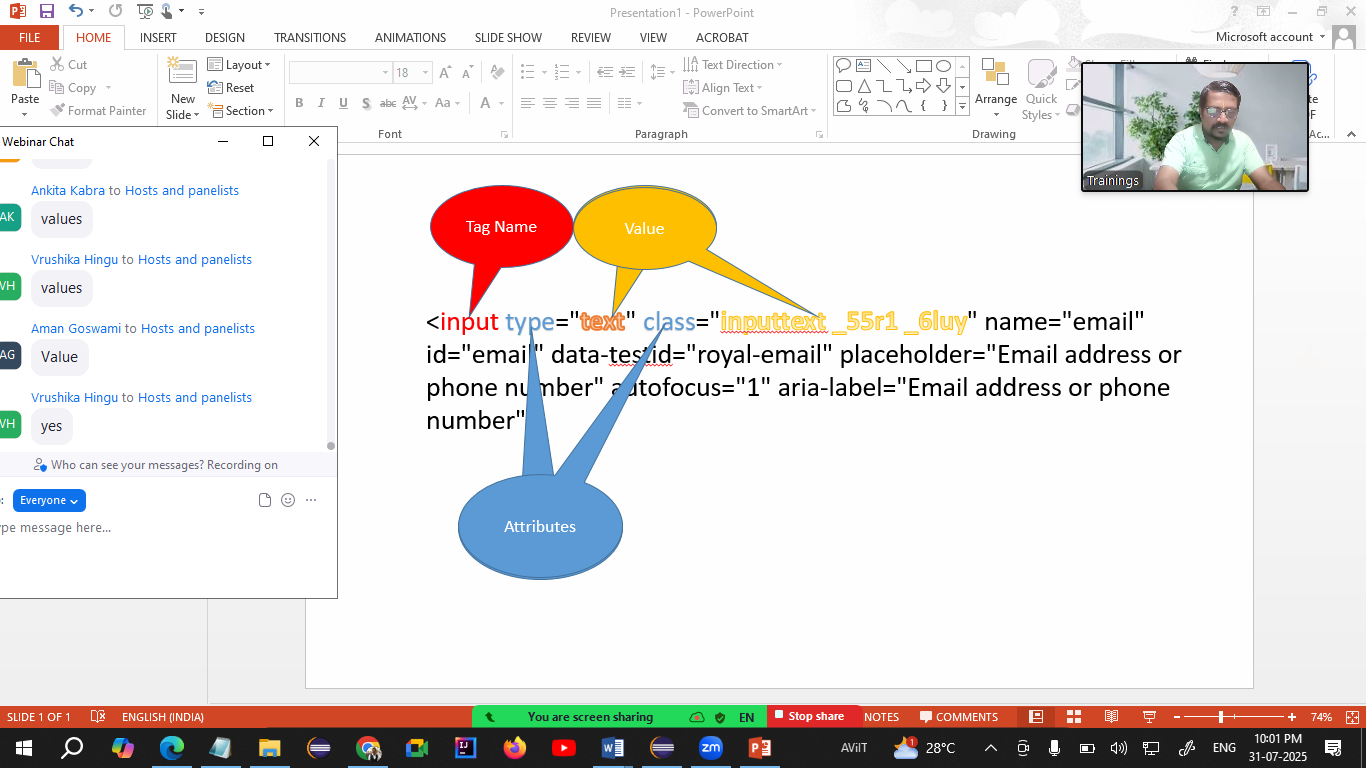
* Open Eclipse
* Select your 1st folder as a workspace using browse button
* Click on Launch button
* Go to File 🡪 New 🡪 Java Project
* Give the name to Project
* Select Java version (Minimum 11)
* **Uncheck Create module-info.java file checkbox**
* Click on Finish
* Create a Package
* Create a class in this package
* Right click on you project (in package explorer) 🡪 Build path 🡪 Configure Build path
* Click on Libraries Tab
* Click on Classpath
* Click on Add Extern JARs… button
* Open your 2nd folder and select the file which you have downloaded in earlier steps. (selenium-server-4.34.0.jar)
* Click on Open button  
  **Make sure that this file is inside the class path**
* Click on Apply and Close

**Methods of WebDriver Interface**

1. Creating Object of WebDriver 🡪 Launch the blank browser window
2. get() 🡪 Launching the Web site via URL. This method requires absolute URL. (Absolute URL starts with http / https)
3. driver.manage().window().maximize() 🡪 Will maximize the browser window.
4. close() 🡪 Close the current browser window which is opened by WebDriver object.
5. getTitle() 🡪 Will return the title of the page which is launched in browser. (String)
6. getCurrentUrl() 🡪 Return the URL of the page which is launched in the browser. (String)
7. getPageSource() 🡪 Returns the rendered HTML of web page. (String)
8. findElement() 🡪 It find and returns the single control from the web page. Reads the first occurrence. (WebElement)
9. findElements() 🡪 It returns multiple controls on the page. (List<WebElement>)
10. getWindowHandles() 🡪 It returns id’s of the windows those are opened by WebDriver object. (Set<String>)
11. quit() 🡪 will close all the windows those are opened by WebDriver object.

**Common Exceptions Occurred during Selenium WebDriver**

1. SessionNotCreatedException 🡪 When the version of Selenium & the version of Browser are not matching
2. InvalidArgumentException 🡪 The URL you passed is not in the correct format.
3. NoSuchElementException 🡪 Selenium us unable to find the control because of
   1. The value of locator is wrong.
   2. The value of locator is dynamic
4. InvalidSelectorException 🡪 The value of locator is in the wrong format.
5. TimeoutException 🡪 The specified time is over in synchronization but your control is not ready yet.
6. SessionTimeoutException 🡪 .get method is unable to load a page within 30 seconds.
7. ElementClickInterceptedException 🡪 When the control is hidden by some another control, and Selenium is unable to click on the control.
8. NoAlertPresentException 🡪 When alert is not present but you still trying to handle an alert.



**WebElement**

* Every control on the page is treated as WebElement.
* WebElement is an interface to hold any control on the page.
* Methods
  + sendKeys() 🡪 Will enter some text in the textbox. Will append the text to existing text in text box.
  + click() 🡪 Will click on any control.
  + getText() 🡪 Will return the text on the control. (String)
  + isSelected() 🡪 Checks that whether the check box / radio button is selected or not (boolean)
  + isEnabled() 🡪 Checks that whether the control is enabled or disabled (boolean)
  + isDisplayed() 🡪 Checks that whether the control is visible or not (boolean)
  + getAttribute() 🡪 Return the value of any attribute. (String)
  + clear() 🡪 Clears the text from text box.

**Locators in Selenium**

Locators are the way of identification of any control (WebElement) on the web page.

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

**CssSelector**

This is the locator via which you can locate any control using any one or multiple attributes of the control.

Type of CssSelector

1. Using Single Attribute  
   Syntax:  
   tagName[attribute=”Value”]  
   input[data-testid="royal-email"]
2. Using Multiple Attributes  
   Syntax:  
   tagName[attribute1=”Value”][attribute2=”Value”]
3. Using Special Characters
   1. ^ 🡪 Starts with  
      tagName[attribute^=”Value”  
      button[id^=”u\_0\_5”]
   2. $ 🡪 Ends With
   3. \* 🡪 Contains

Pattankodoli Bus stand 🡪 Take right turn 🡪 Hupare Nagar 🡪 Water Tank 🡪 Lane No 9 🡪 House No 1128 (Ankush Home)

XPath (Xml Path)

1. Absolute XPath  
   starts with html
2. Relative XPath
   1. Take a reference of parent tag
   2. Using specific control (tag)

**Handling Dropdown List**

* If the control is having <select> tag then only the control is treated as drop down list.
* **Select Class** is used to handle dropdown list / List box.
* You can perform operations on dropdown list only via Select class.

Operations

1. Display the selected country
2. Display total no of countries
3. Display list of all the countries.
4. Select Poland from the list
5. Display selected country.

**Methods of Select Class**

1. getFirstSelectedOption() 🡪 Return the selected element / option from the drop down list. (WebElement)
2. getOptions() 🡪 Return list of all the elements / options from drop down list (List<WebElement>)
3. selectByContainsVisibleText()
4. selectByVisibleText()
5. selectByValue()
6. selectByIndex()
7. getAllSelectedOptions() 🡪 Return list of all the selected options from list box (List<WebElement>)
8. isMultiple() 🡪 Will check that whether the control allows to select multiple options or not (boolean)

**Synchronization (Waits in Selenium)**

Synchronization is process of adjusting speed of tool with speed of application.

1. Thread.sleep() 🡪 Will pause the execution of the script for specified milliseconds.
   1. Take a mandatory delay
   2. Applicable to single statement
2. Implicit Wait
   1. It is applicable throughout the script
   2. It doesn’t take mandatory delay
3. Explicit Wait (WebDriverWait class)
   1. It doesn’t take mandatory delay
   2. It is applicable to single statement only.
   3. We can handle some conditions like element to be clickable, alert to be displayed, visibility of element
4. Fluent Wait 🡪 This is next version of explicit wait, but this method is able to handle exception.
   1. It doesn’t take mandatory delay
   2. It is applicable to single statement only.
   3. We can handle some conditions like element to be clickable, alert to be displayed, visibility of element
   4. Having ability to handle some specific exception  
      w - withTimeout  
      i - ignoring  
      p - pollingEvery  
      u – until
5. PageLoadTimeout 🡪 We can add some more time for loading the page to avoid SessionTimeoutException.

**Handling Table**

1. Display all the headers
2. Display total no of rows
3. Display any row randomly

**JavascriptExecutor Interface**

1. To scroll the page vertically, horizontally.
2. To click on the control which is behind any ad / control (Hidden by some another control)

**Handling Alerts**

Alert is a small window that

1. Normally appears on the center top of browser
2. If this window is visible, you are not allowed / able to perform any actions on the page
3. You are not able to inspect this window

**Alert** interface is used to handle alert in Selenium WebDriver.

Methods

1. getText() 🡪 Return the text on the Alert
2. accept() 🡪 Click on Ok button
3. dismiss() 🡪 Click on Cancel button
4. sendKeys() 🡪 Will enter the data on Alert (Prompt box)

**Mouse Actions**

1. Click
2. Right Click
3. Double Click
4. Hover
5. Drag and Drop

**Actions** class from Selenium is used to perform all above actions.

Methods

1. moveToElement() 🡪 Hovers the mouse on the control.
2. perform() 🡪 Needs to call after every mouse action to perform the said action.
3. contextClick() 🡪 Performs right click.
4. doubleClick() 🡪 Performs double click on the control.
5. dragAndDrop() 🡪 Perform drag and drop action. Source will be dragged and dropped on the target.

**TestNG (Test Next Generation)**

It is testing framework

**Framework:** It is set of rules, guidelines that will make your testing more easily.

**Advantages**

* Reduces testing time / code
* Combination of multiple tests
* Allows to execute single / multiple tests
* Set the priority to test cases
* Uses annotations
  + @Test
  + @BeforeTest
  + @AfterTest
  + @BeforeMethod
  + @AfterMethod
  + @Parameters
* Data driven testing using @DataProvider
* Automatic Report
  + Normal Report
  + HTML Report
* Execute / skip single test / multiple tests
* Parallel Execution
* Execute tests via groups
* Parameterization
* Implements some frameworks
  + Linear framework
  + Modular Framework
  + Keyword driven framework
  + Page Object Model (POM)
  + Data Driven Framework
  + Hybrid Framework

**TestNG annotations**

1. @Test 🡪 The method which is having @Test annotation will be treated as test case.  
   Tests will get executed in its alphabetical order  
   priority = 1 🡪 Will set the priority of test case  
   enabled = false 🡪 Will disable the test from execution
2. @BeforeTest 🡪 This method will get executed **only once before executing 1st test case.**
3. @AfterTest 🡪 This method will get executed **only once after executing last test.**
4. @BeforeMethod 🡪 This method will get executed **before every test case.**
5. @AfterMethod 🡪 This method will get executed **after every test case.**
6. @DataProvider 🡪 This method sends the data to @Test method

BeforeTest

BeforeMethod

Flipkart

AfterMethod

BeforeMethod

Myntra

AfterMethod

BeforeMethod

Amazon

AfterMethod

BeforeMethod

Meesho

AfterMethod

AfterTest

**Points to be noted about before and after methods (Configuration methods)**

1. **They can appear anywhere in the script (Sequence doesn’t matter)**
2. **They need not to be in a pair**

**Data Driven Testing**

You can execute single test for multiple times with multiple data set.

**Assertion**

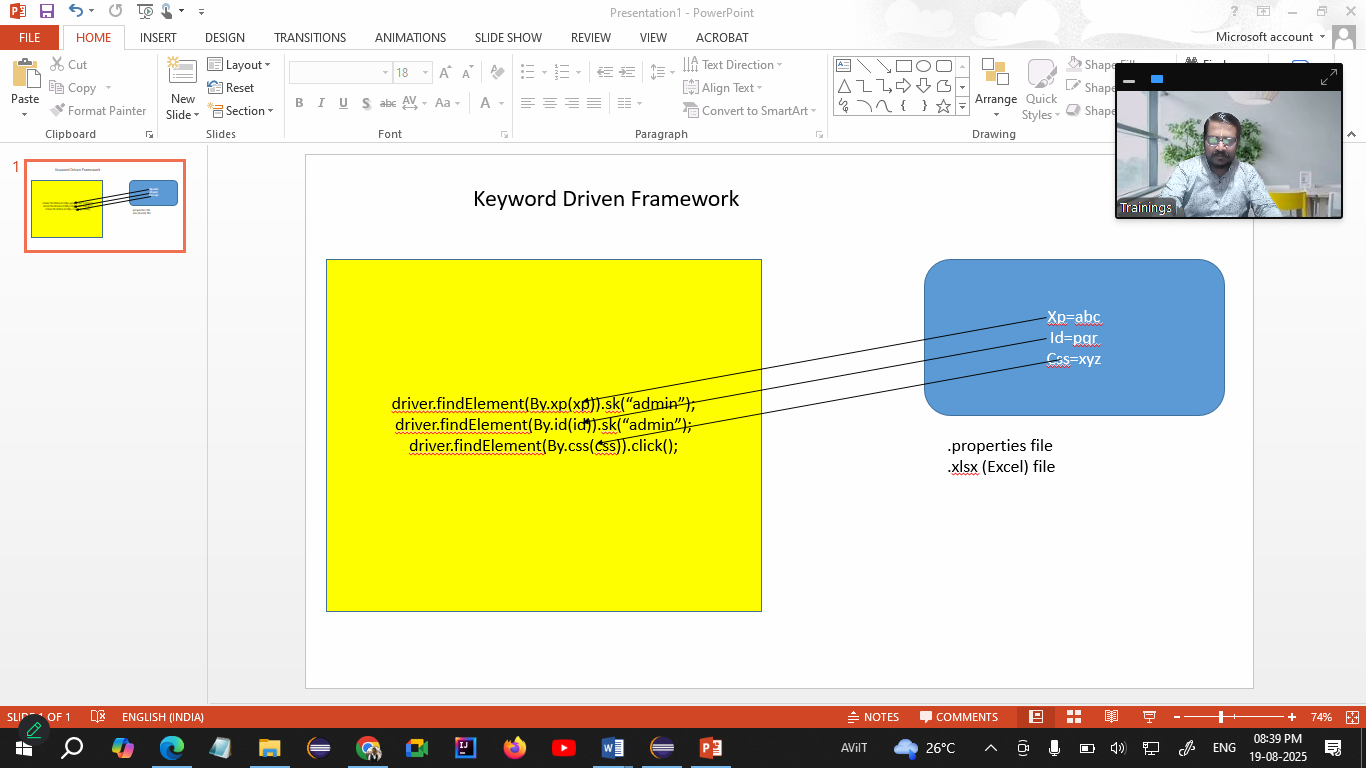
Process of marking any test case as pass or fail in report.

**Modular Framework**

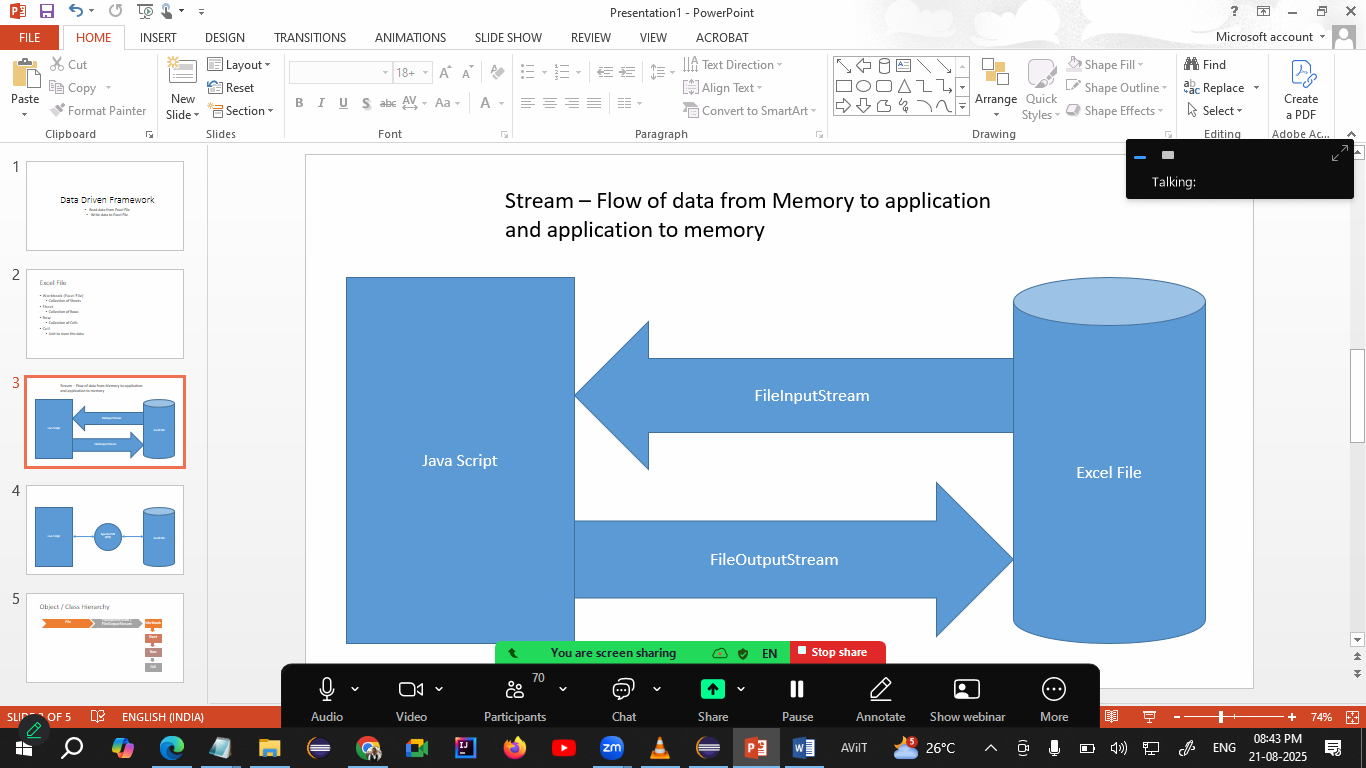
* Allows to execute / skip single / multiple tests.
* Executing the tests via XML file.

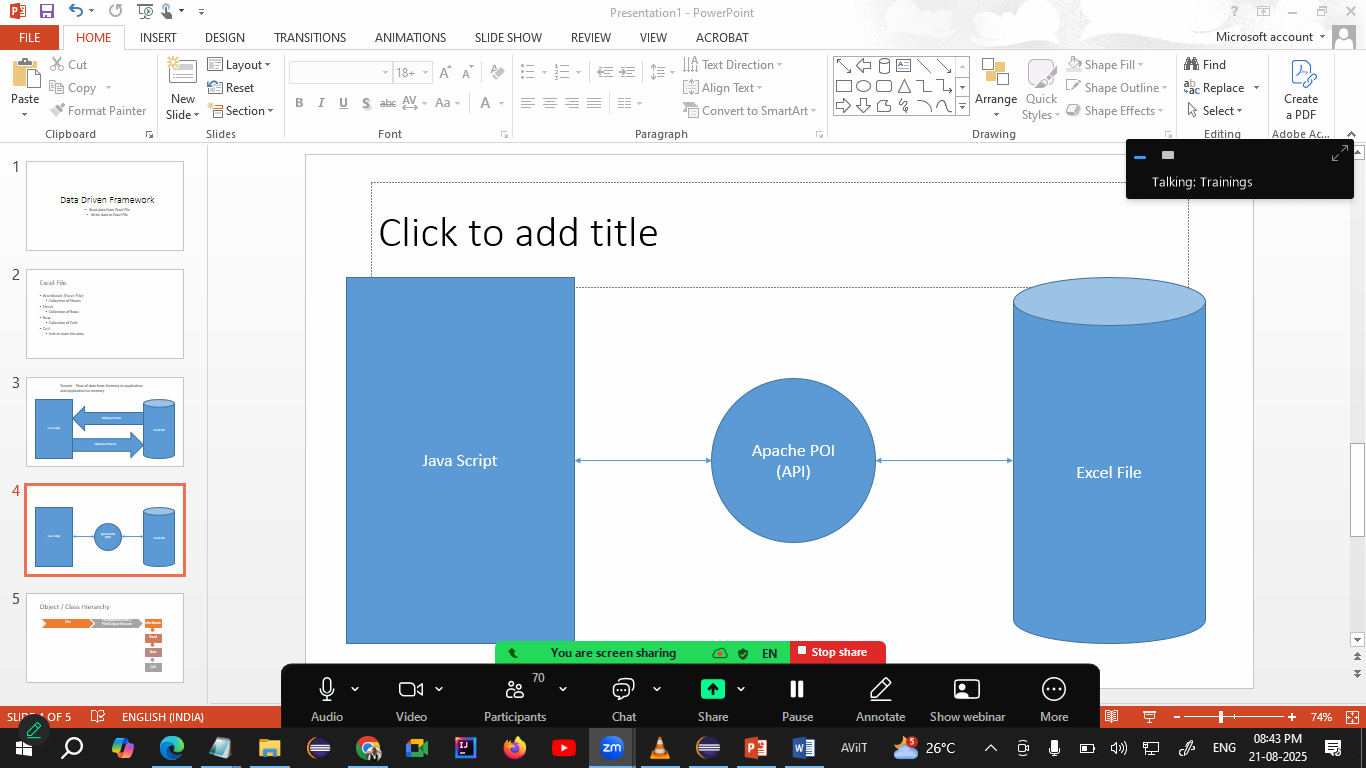
**Points to be noted while creating XML file**

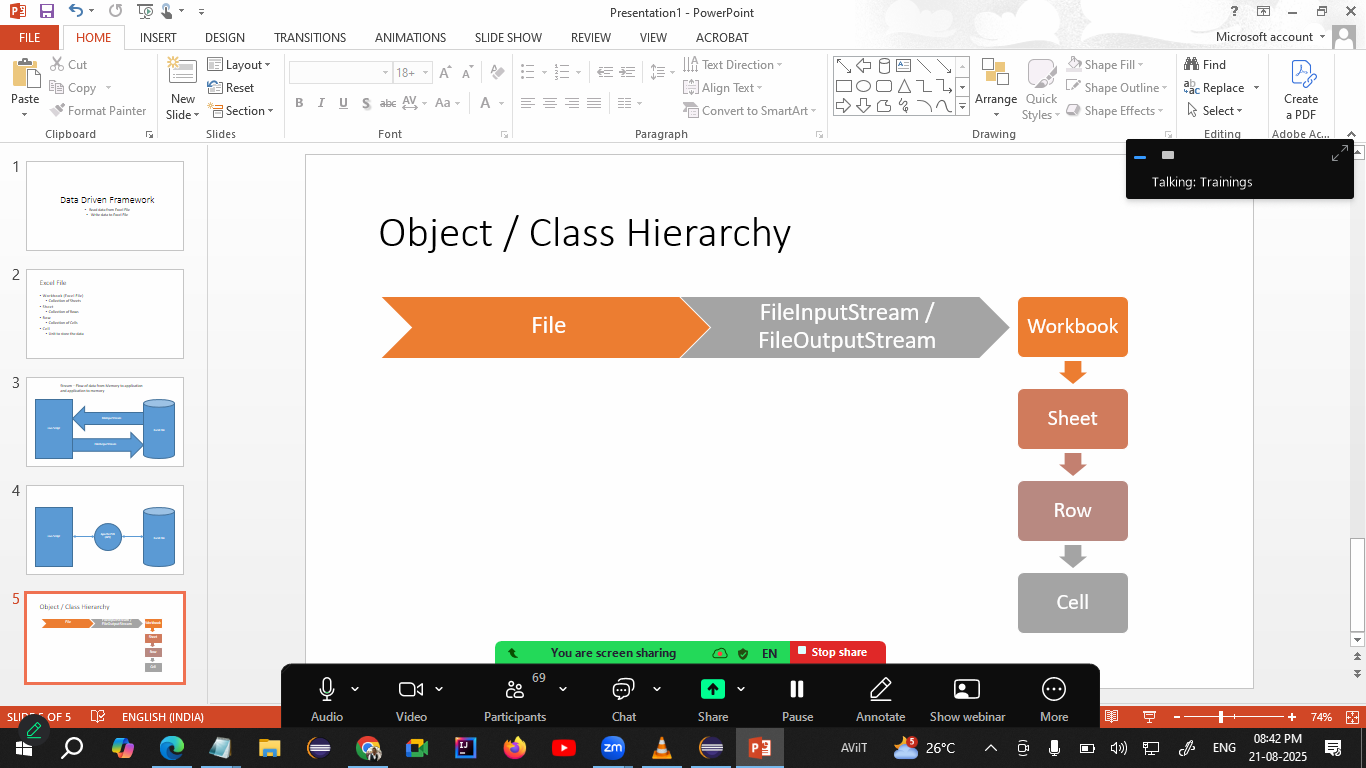
1. All the tags are pre-defined
2. The sequence of tag cannot be altered.



Data Driven Framework







Apache POI Configuration

Launch <https://archive.apache.org/dist/poi/release/bin/>

Download last 4th file [poi-bin-5.2.3-20220909.zip](https://archive.apache.org/dist/poi/release/bin/poi-bin-5.2.3-20220909.zip)

Copy paste this file to 2nd folder (Selenium Jar File)

Extract this (.zip) file

Observe the folder structure

8 jar files in root folder

6 jar files in lib folder

7 jar files in ooxml folder

You need to add all these 21 files to your project. Follow the steps which we did in 2nd lecture.

* Right click on you project (in package explorer) 🡪 Build path 🡪 Configure Build path
* Click on Libraries Tab
* Click on Classpath
* Click on Add Extern JARs… button
* Open your 2nd folder and select all 21 files one by one folder
* Click on Open button  
  **Make sure that this file is inside the class path**
* Click on Apply and Close

XSSF 🡪 handling .xlsx files

HSSF 🡪 handling .xls files

**Maven – Build Management Tool**

* It is Apache product
* Can be used by both developers and testers
* Can share a single project by both
* Configuration is very easy via pom.xml file
* Open source
* Uses dependencies

**Creating Maven Project**

* File Menu 🡪 New 🡪 Maven Project
* Select the 1st checkbox (create a simple project. Skip archetype)
* Click on next
* Give GroupID and ArtifaceId (Both should be same)
* Click on Finish

**For Execution you have to add**

* **Maven Compiler Plugin**
* **Maven Surefire Plugin**

**BDD Approach (Behavior Driven Development)**

* BA, Developer & QA team will gather the requirements from client
* Requirements will be shared with QA team.
* QA team will create **Feature file (1st Component of BDD).**
* Feature file is collection of test scenarios
  + Will be shared with developer
  + Will be shared with Manual tester
  + Will be shared with automation tester
* Automation tester will execute this feature file
* After executing feature file you will get **Step Definition / Glue code (2nd component of BDD)**
* Glue code contains the automation script
* **Runner class (3rd component of BDD)** will help to execute the feature file / glue code

**Cucumber is tool to implement BDD Approach**

**Creating Cucumber project**

1. Click on File Menu 🡪 New 🡪 Maven Project
2. Select 3rd check box (Add project to working set)
3. Click on Next
4. Add io.cucumber in filter
5. Click on Next
6. Add groupId & artifactID
7. Click on Finish
8. Delete the pre-defined package from src/test/java
9. Launch <https://github.com/cucumber/cucumber-java-skeleton/commit/d7249b50c570816eba27ce94557e1de7e9b0f97>
10. Copy the code from <properties> till </dependencies> (line no 11 to 41)
11. Open pom.xml file
12. Delete the code from <properties> till </dependencies> (line no 12 to 67)
13. Change the java version in following line  
    <java.version>1.8</java.version>
14. Delete following lines

<cucumber.version>6.8.2</cucumber.version>

<maven.compiler.version>3.8.1</maven.compiler.version>

<maven.surefire.version>2.22.2</maven.surefire.version>

1. Save this file

**Components of BDD**

* Feature File
* Collection of Test Scenarios
* Created using Gherkin language
* Created using Keywords
  + Feature: 🡪 Represents requirement
  + Scenario: 🡪 Represents Test Objective
  + Given 🡪 Represents Pre-Requisite
  + When 🡪 Represents steps to be perform
  + And 🡪 Used to combine multiple When statements
  + Then 🡪 Represents Expected result
  + Background: 🡪 Used to avoid repeated Given statement for multiple Scenarios.
  + Examples: 🡪 Used to implement Data Driven Testing
  + Scenario Outline: 🡪 Used to combine Examples. (Data Driven Testing)

Requirement 🡪 Validate Google Home Page

1. Open Google
2. Read the title
3. Title should be Google

Feature File

Feature: Google Home Page

Scenario: To validate Google Home page

Given Open Google

When Read the title

Then Title should be Google

Requirement – Validate Google Search

1. Open Google
2. Enter the text in search box
3. Hit Enter
4. Valid search result should display

Feature File

Feature: Validate Google Search

Scenario: To Validate search functionality on Google

Given Open Google

When Enter word to be searched

And Hit Enter

Then Valid search result should be displayed

**Tags in Cucumber**

* Tags can be used to execute or skip single or multiple scenarios
* Tags are specified in .feature file
* Tags are created using @

**Hooks in Cucumber**

* The methods those get executed before and after every scenario
* There are 2 hooks
  + @Before
  + @After
* Hooks are not the part of feature file.
* It can added either in Step definition class or even you create a separate class for the hooks