**Day - 1**

**Introduction on Selenium**

Selenium is an open-source framework for automated browser testing and web application automation, offering a suite of tools to interact with web browsers programmatically

**Different kind of Selenium components**

Selenium comprises four main components: Selenium IDE (Integrated Development Environment) for record-and-playback, Selenium WebDriver for programmatic browser control, Selenium Grid for parallel test execution, and Selenium Standalone Server to facilitate communication between the WebDriver and browsers

**Selenium Architecture**

Selenium's architecture revolves around a browser driver, a client library, and the WebDriver protocol, enabling automated testing and interaction with web browsers

**Selenium Installation**

Installation of Selenium web driver, making sure all Environmental variables are set and installing selenium library in IDE.

**Locators** :

What is locators, deep dive on different kind of locators

Selenium locators are tools that enable precise identification of web elements for automation, using strategies like ID, class name, XPath, CSS selectors, and more.

**Day 2 –**

**Web Element Commands: Edit Box, Button, Check box, Radio Button.**

A webpage is composed of various web elements that users interact with, and Selenium

provides a way to automate interactions with these elements. Common web elements include buttons, text fields, checkboxes, radio buttons, dropdowns, links, and images. Selenium's Web Element class represents these elements, offering methods to interact with and extract information from them

Selenium Web Element commands serve as the bridge between automation scripts and web elements, allowing for interactions like clicks, text input, and checkbox, radio button selections. Through methods such as **click()**, **sendKeys().**

**Handling DropDown, Multiple Checkboxes, Radio buttons and Actions class.**

The Select class in Selenium simplifies dropdown handling by providing methods for easy option selection, enabling efficient automation of dropdown menus

The Actions class in Selenium facilitates complex user interactions, making it invaluable for handling dropdowns through methods like moveToElement()

In Selenium, the **findElements()** method is used to locate multiple web elements on a page based on a specified locator strategy,

**Day 3 -**

**Handling Popups using Alert, Working with Web Elements.**

Pop-ups in web applications are additional browser windows that appear dynamically, often requiring special handling in automation scripts for effective testing using Selenium.

**Handling IFrames**

Iframes (Inline Frames) in web development are HTML elements that allow embedding one HTML document within another.

In Selenium, handle frames by switching the driver's focus using **SwitchTo()** before interacting with elements inside

**Handling Window Handels & Multiple Tabs.**

Handle multiple windows and tabs in Selenium using **getWindowHandles()** to retrieve window handles, then switch between them using **switchTo().window() method.**

**Day – 4**

**Wait commands & Java Script Executor**

These wait strategies are essential for handling synchronization issues in test automation, ensuring that the script waits for the page to load or elements to become intractable before performing actions. Each wait type caters to different scenarios and helps improve the reliability of automated tests.

**Implicit Wait** :

Implicit wait in Selenium is a global setting that instructs the WebDriver to wait for a specified amount of time during each find element command before throwing a NoSuchElementException.

**Explicit Wait** :

Explicit wait in Selenium is a targeted approach where the WebDriver waits for a specific condition to be met before proceeding, enhancing synchronization in test automation.

**JavaScriptExecutor**

in Selenium enables the execution of JavaScript code within the WebDriver, providing a powerful tool for manipulating the DOM and performing advanced interactions during test automation

**Drag & Drop** : Selenium supports drag and drop through the Actions class, enabling the simulation of mouse movements for this interaction.

**Broken Links** :

Identify and handle broken links in Selenium by checking HTTP response codes or using link validation techniques, ensuring web application reliability

**Taking Screenshot** :

Capture screenshots in Selenium using the TakesScreenshot interface, providing visual evidence for test execution and debugging.

**File Upload & Download :**

Automate file uploads in Selenium by interacting with the file input element and providing the file path for seamless testing of upload functionalities

**Day – 5**

**Introduction to Data Driven Testing using Apache POI & implementation.**

Data-driven testing in Selenium using Apache POI involves reading test data from external Excel files, enhancing test robustness and flexibility by separating data from the test scripts

**Introduction to TestNG annotations**

TestNG is a Java testing framework that facilitates efficient test configuration, parallel execution, and detailed reporting, offering annotations for easy test case management

**Difference between Junit & TestNG.**

JUnit is a widely used Java testing framework for unit testing, providing annotations to define and execute test cases, ensuring code reliability and maintainability