EXPERIMENT NO.: 10.

Install and Explore Selenium for automated testing

AIM: To install the necessary software and explore Selenium WebDriver for automated web testing. This includes setting up the Java Development Kit (JDK), Selenium WebDriver, browser drivers, and an Integrated Development Environment (IDE), and then writing and executing a basic test script.

DESCRIPTION:

- Selenium is a widely-used open-source framework for automating web browsers. It provides a suite of tools for automating web applications for testing purposes but is also used for web scraping and automating repetitive tasks
- Selenium supports a variety of programming languages through the use of language-specific bindings or drivers. The languages supported by Selenium include java, C#, python, Ruby, Javascript, perl.
- Selenium test scripts can be written in any of the supported programming languages and executed in modern web browsers.
- Selenium supports a wide range of web browsers, including:
  - **Google Chrome**
  - **Mozilla Firefox**
  - **Internet Explorer**
  - **Microsoft Edge**
  - **Safari**

A simple example demonstrating how to use Selenium WebDriver with Java to open Google's homepage.

Step1: Install JDK and set up environmental variables.

Step2: Install selenium web driver

Selenium WebDriver is a set of APIs that allow you to control web browsers.

Go to https://www.selenium.dev/downloads/ and download java driver under selenium clients and webdriver language bindings. Extract the zip file and place in C:\Drivers\

Step3: Install browser driver.

Each supported browser has a corresponding driver that Selenium WebDriver uses to communicate with the browser. For example:

- ChromeDriver: For Google Chrome
- GeckoDriver: For Mozilla Firefox
- IEDriverServer: For Internet Explorer
- SafariDriver: For Safari.

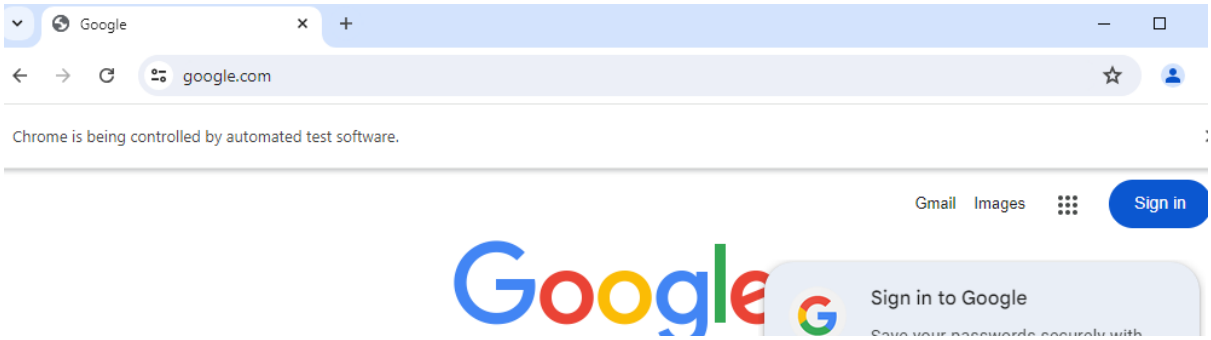
For chrome driver, check the google chrome version and go to https://googlechromelabs.github.io/chrome-for-testing/ . Download the stable release, Extract the zip file and place in C:\Drivers\selenium-java-<version>\

Step4: Create a simple test application test.java

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Test {
    public static void main(String[] args) {
        // Set the path to the ChromeDriver executable
        System.setProperty("webdriver.chrome.driver", "C:\\Drivers\\selenium-java-4.23.0\\chromedriver-win64\\chromedriver.exe");

        // Initialize the ChromeDriver
        WebDriver driver = new ChromeDriver();

        // Open a webpage
        driver.get("https://www.google.com");

        // Print the title of the page
        System.out.println("Page title is: " + driver.getTitle());

        // Close the browser
        driver.quit();
    }
}
```

Step5: Compile and run the program with the following commands

```
C:\Users\kuppa\testing>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Test.java

C:\Users\kuppa\testing>java -cp "C:\Drivers\selenium-java-4.23.0\*;." Test
Page title is: Google
```



EXPERIMENT NO.: 11.
Write a simple program in JavaScript and perform testing using Selenium

AIM: To write a selenium test case for a simple JavaScript program that increments a counter on a webpage. The javaScript program adds a counter functionality to a button. The goal is to test this functionality using Selenium.

Description:

Step1: Write JavaScript Code: Develop a web application with JavaScript that performs certain tasks (e.g., incrementing a counter).

Step2: Write Selenium Test in Java: Use Selenium to write a test in Java that interacts with the web page, triggering JavaScript actions and verifying outcomes.

Step3: Download the following jar files and place them in c:\Drivers

- **JUnit:** Defines and runs test cases in Java, checking that the Selenium WebDriver interactions produce the expected outcomes. Install Junit jar and place it in c:\Drivers
- **Hamcrest:** Provides expressive assertion methods for verifying conditions in your tests. Install Hamcrest jar and place it in c:\Drivers

counter.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Simple JavaScript Program</title>
</head>
<body>
    <p id="output">0</p>
    <button id="increment-button">Increment</button>
    <script>
        const output = document.getElementById("output");
        const incrementButton = document.getElementById("increment-button");
        let count = 0;
        incrementButton.addEventListener("click", function() {
            count += 1;
            output.innerHTML = count;
        });
    </script>
</body>
</html>
```

Main.java

```java
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

public class Main {
    private WebDriver driver;

    @Before
    public void setUp() {
        System.setProperty("webdriver.chrome.driver", "C:\\Drivers\\selenium-java-4.23.0\\chromedriver-win64\\chromedriver.exe");
        driver = new ChromeDriver();
    }

    @Test
    public void testIncrementButton() throws InterruptedException {
        driver.get("file:///C:/Users/sample/counter.html");
        driver.findElement(By.id("increment-button")).click();
        String result = driver.findElement(By.id("output")).getText();
        assert result.equals("1") : "Expected output to be 1, but got " + result;
        Thread.sleep(5000);
    }

    @After
    public void tearDown() {
        driver.quit();
    }
}
```

Step4: compile and run the program.

```
C:\Users\sample>javac -cp "C:\Drivers\selenium-java-4.23.0\*" Main.java

C:\Users\sample>java -cp "C:\Drivers\selenium-java-4.23.0\*;." org.junit.runner.JUnitCore Main
JUnit version 4.13.2
.
Time: 10.976

OK (1 test)
```

EXPERIMENT NO.: 12.
Develop testcases for the containerized application using selenium.

Aim: To develop testcases for the containerized application using selenium

Description: Selenium is a powerful tool for automating web tests. Docker is a containerization platform that allows us to package an application and all its dependencies into a single, isolated container.
By containerizing Selenium with Chrome, you can streamline your testing process, simplify CI/CD pipelines.

The following are the steps to run Selenium tests in Docker using a standalone Chrome instance.

Step 1: Pull the Selenium Standalone Chrome Docker Image

First, ensure you have Docker installed on your system. Then, pull the Selenium standalone Chrome image:

```
C:\Users\seleniumtestindocker>docker pull selenium/standalone-chrome
Using default tag: latest
latest: Pulling from selenium/standalone-chrome
Digest: sha256:a6dbdd1eb16f67e9e927f9c48a29d480250b3dd1364a8c9d9d0d354f2d86c230
Status: Image is up to date for selenium/standalone-chrome:latest
docker.io/selenium/standalone-chrome:latest
```

```
C:\Users\seleniumtestindocker>docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
mypythonapp                   latest    5ce422723ec7   19 minutes ago   125MB
selenium/standalone-chrome    latest    a0672e3f5bb0   4 weeks ago      1.17GB
```

Step 2: Run the Selenium Standalone Chrome Container

Start the Selenium standalone Chrome container:

```
C:\Users\seleniumtestindocker>docker run -d -p 4444:4444 --shm-size=2g selenium/standalone-chrome
8d000dcda78d61480a4d81c43e3aedf1971d7ca7334561b1382d8d8fe7af66bf
```

This command will:

- Run the container in detached mode (`-d`).
- Map port 4444 of the host to port 4444 of the container (`-p 4444:4444`).
- Allocate 2 GB of shared memory (`--shm-size=2g`) to avoid some common browser crashes.
- Use the `selenium/standalone-chrome` image.

Step 3: Create a Selenium Test Script

1. **Create the test script `GridExample.java`:**

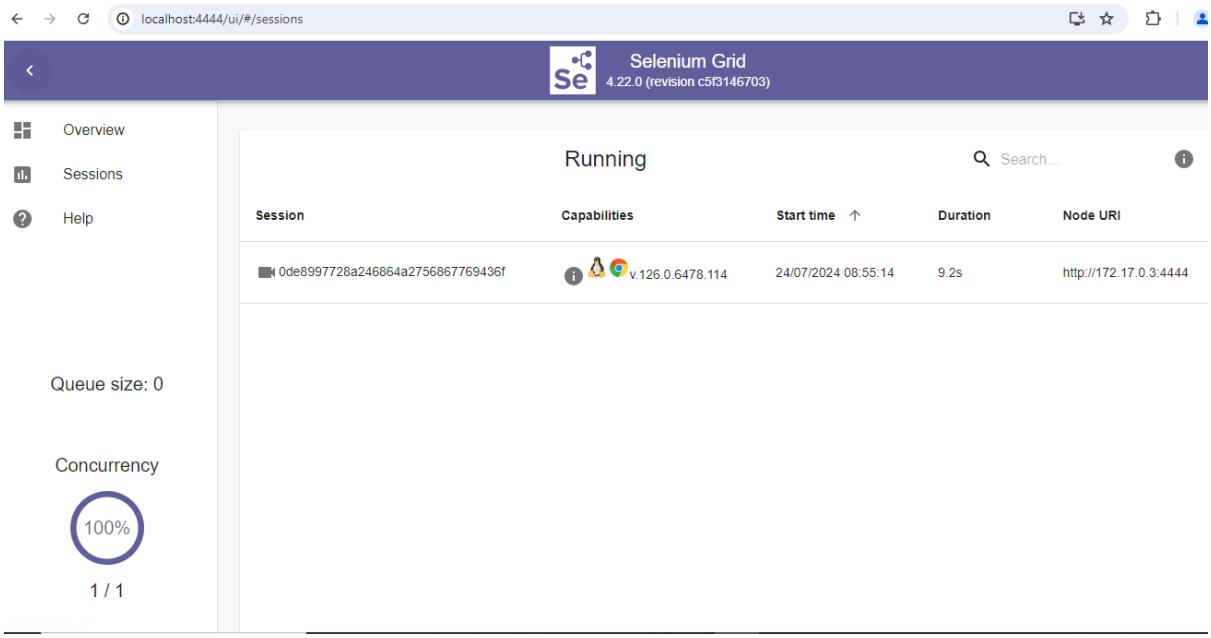
```
package com.example;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.remote.RemoteWebDriver;

import java.net.URL;

public class GridExample {
    public static void main(String[] args) {
        WebDriver driver = null;
        try {
            // Define Chrome options
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--ignore-ssl-errors=yes");
            options.addArguments("--ignore-certificate-errors");

            // Replace "localhost" with the actual address of your Selenium Grid hub
            URL hubUrl = new URL("http://localhost:4444/wd/hub");

            // Create a RemoteWebDriver instance pointing to the hub
            driver = new RemoteWebDriver(hubUrl, options);

            // Maximize the window size
            driver.manage().window().maximize();
            Thread.sleep(10000); // Just for demonstration

            // Navigate to browserstack.com
            driver.get("https://www.browserstack.com/");
            Thread.sleep(10000); // Just for demonstration

            // Click on the "Get started for free" button
            driver.findElement(By.linkText("Get started free")).click();
            Thread.sleep(10000); // Just for demonstration
        } catch (Exception e) {
```

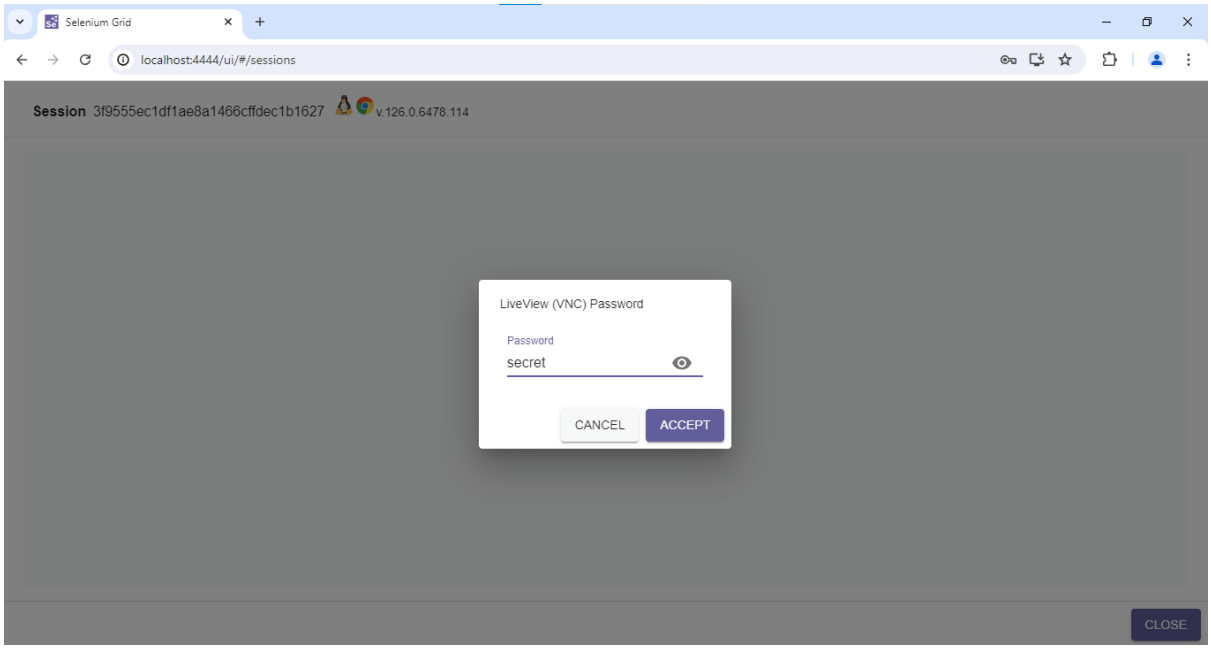
```
            e.printStackTrace();
        } finally {
            if (driver != null) {
                driver.quit();
            }
        }

        // Ensure the JVM exits properly
        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            System.out.println("Shutdown hook ran!");
        }));
    }
}
```

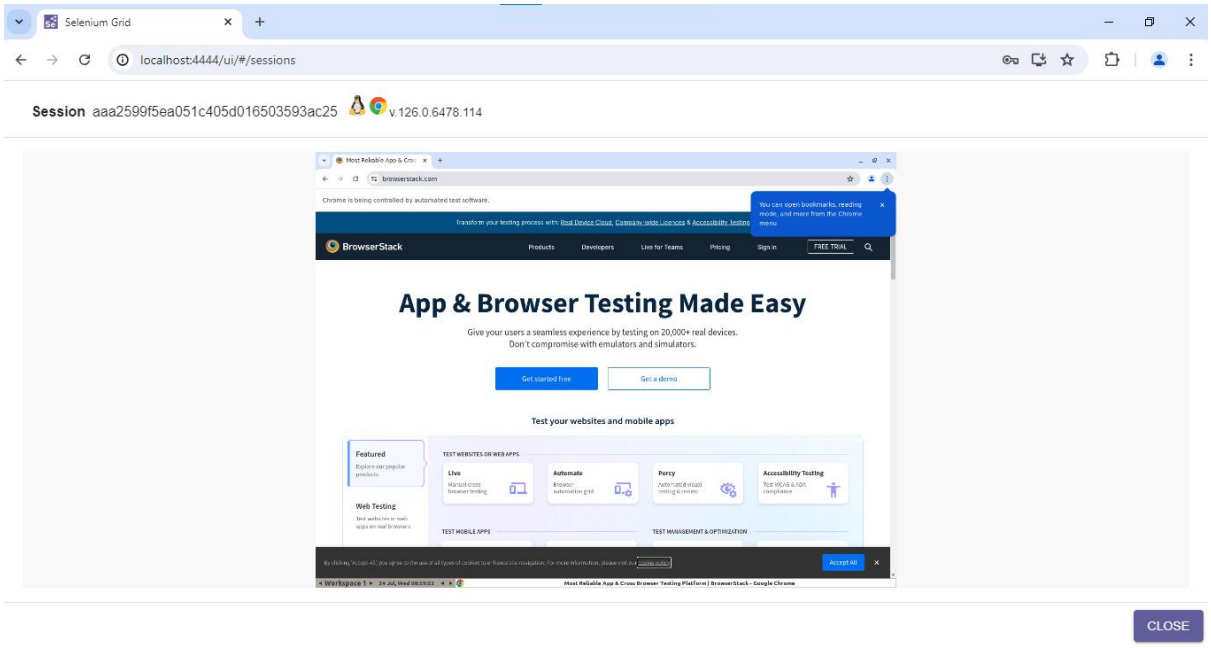
Step4: Build and Run Your Java Test Code

```
C:\Users\seleniumtestindocker>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:selenium-java-docker >------------------
[INFO] Building selenium-java-docker 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ selenium-java-docker ---
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\seleniumtestindocker\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ selenium-java-docker ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  19.239 s
[INFO] Finished at: 2024-07-24T14:23:53+05:30
[INFO] ------------------------------------------------------------------------
```

```
C:\Users\seleniumtestindocker>mvn exec:java -Dexec.mainClass="com.example.GridExample"
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< com.example:selenium-java-docker >------------------
[INFO] Building selenium-java-docker 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- exec-maven-plugin:3.0.0:java (default-cli) @ selenium-java-docker ---
```


localhost:4444/ui/#/sessions
Selenium Grid
4.22.0 (revision c5f3146703)
Overview
Sessions
Help
Running
Search...
Session
Capabilities
Start time
Duration
Node URI
0de8997728a246864a2756867769436f
v.126.0.6478.114
24/07/2024 08:55:14
9.2s
http://172.17.0.3:4444
Queue size: 0
Concurrency
100%
1 / 1


Selenium Grid
localhost:4444/ui/#/sessions
Session 3f9555ec1df1ae8a1466cffdec1b1627
v.126.0.6478.114
LiveView (VNC) Password
Password
secret
CANCEL
ACCEPT
CLOSE


Selenium Grid
localhost:4444/ui/#/sessions
Session aaa2599f5ea051c405d016503593ac25 v.126.0.6478.114
Chrome is being controlled by automated test software.
You can open bookmarks, reading mode, and more from the Chrome menu
BrowserStack
Products
Developers
Live for Teams
Pricing
Sign in
FREE TRIAL
App & Browser Testing Made Easy
Give your users a seamless experience by testing on 20,000+ real devices.
Don't compromise with emulators and simulators.
Get started free
Get a demo
Test your websites and mobile apps
Featured
Web Testing
TEST WEBSITES OR WEB APPS
Live
Automate
Percy
Accessibility Testing
TEST MOBILE APPS
TEST MANAGEMENT & OPTIMIZATION
Accept All
Most Reliable App & Cross Browser Testing Platform | BrowserStack - Google Chrome

CLOSE