**1. Demonstrate the features of Selenium**

Features of Selenium

* **Open-Source:**

Selenium is a freeware and a portable tool. It has no upfront direct costs involved. The tool can be freely downloaded and the support for it is freely available, as it is community-based.

* **Supports languages:**

Selenium supports a range of languages, including Java, Perl, Python, C#, Ruby, Groovy, Java Script, etc. It has its own script, but it doesn’t limit it to that language. It can work with various languages and whatever the developers/testers are comfortable with.

* **Supports Operating Systems:**

Selenium operates across and supports multiple Operating Systems, (OS) like Windows, Mac, Linux, UNIX, etc. With Selenium Suite of solutions, a tailored testing suite can be created over any platform and then executed on another one. For instance, you can create test cases using Windows OS and run it with ease on a Linux-based system.

* **Supports multiple browsers:**

Selenium provides support across multiple browsers, namely, Internet Explorer, Chrome, Firefox, Opera, Safari, etc. This becomes highly resourceful while executing tests and testing it across various browsers simultaneously.

The browsers supported by the Selenium packages are:

* Selenium IDE can be used with Firefox as a plug-in.
* Selenium RC and Webdriver supports diverse browsers, such as Internet Explorer.
* Supports programming languages and frameworks

Selenium integrates with programming languages and various frameworks. For instance, it can integrate with ANT or Maven type of framework for source code compilation. Furthermore, it can integrate with the TestNG testing framework for testing applications and reporting purposes. It can integrate with Jenkins or Hudson for Continuous Integration (CI) and can even integrate with other Open-Source tools to support other features.

* **Tests across devices**

Selenium Test Automation can be implemented for mobile web application automation on Android, IPhone, and Blackberry. This can help in generating necessary results and addresses issues on a continuous basis.

* **Constant updates**

Selenium support is community-based and active community support enables constant updates and upgrades. These upgrades are readily available and do not require specific training. This makes Selenium resourceful and cost-effective as well.

* **Loaded Selenium Suites**

Selenium is not just a singular tool or utility, it is a loaded package of various testing tools and so is referred to as a Suite. Each tool is designed to cater to different testing needs and requirements of test environments.

Additionally, Selenium comes with capabilities to support Selenium IDE, Selenium Grid, and Selenium Remote Control (RC).

* **Ease of implementation**

Selenium offers a user-friendly interface that helps create and execute tests easily and effectively. Its open-source features help users to script their own extensions which makes it easy to develop customized actions and even manipulate at an advanced level. Tests run directly across browsers and users can watch while the tests are being executed. Additionally, Selenium’s reporting capabilities are one of the reasons for being chosen, as it allows testers to extract results and take follow-up actions.

* **Reusability and Add-ons**

Selenium Test Automation Framework uses scripts that can be tested directly across multiple browsers. Concurrently, it is possible to execute multiple tests with Selenium, as it covers almost all aspects of functional testing by implementing add-on tools that broaden the scope of testing.

**2. Demonstrate how Selenium web driver is installed and integrated in Eclipse.**

**Development Environment**

* Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
* JavaDevelopment Kit Version 8

This guide has mainly three subsections, namely:

4.2.1 Downloading Selenium Standalone Server jar

4.2.2 Launching Eclipse and creating a Java project

4.2.3 Configuring WebDriver with Eclipse

**Step 4.2.1:** Downloading Selenium Standalone Server jar

* Selenium is already installed in your practice lab. (Refer QA to QE: Lab Guide - Phase 1)

**Step 4.2.2:** Launching Eclipse and creating a Java project

* Launch the Eclipse and create a Workspace.
* Create Project:

Click on File -> New -> Java Project.

**Step 4.2.3:** Configuring WebDriver with Eclipse

* Add selenium standalone server jars.
* Right-click on Project -> select Properties -> Select Java Build Path.
* Navigate to the Libraries tab and click on the Add External Jars button.
* Add selenium standalone server Jar files.
* Click on the Apply and Close button.
* In Eclipse, it looks like the screenshot below:



3. Demonstrate how elements are located using Selenium WebDriver.

**package** demo;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** VerifyTitle {

**public** **static** **void** main(String[] args) {

//1) open the browser

ChromeDriver driver = **new** ChromeDriver();

//2) navigate to application

//object.method()

driver.get("https://facebook.com");

//3)Verify the visitor on the page sees the title - 'Facebook - log in or sign up'

String expectedTitle = "Facebook – log in or sign up";

String actualTitle = driver.getTitle();

System.***out***.println(expectedTitle);

System.***out***.println(actualTitle);

**if**(expectedTitle.equals(actualTitle)) {

System.***out***.println("Test Case Passed");

}

**else** {

System.***out***.println("Test Case Failed");

}

// 4)Close the browser

driver.quit();

}

}

4. Demonstrate how elements are located through CSS and XPath.

package demo;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

public class VerifyFromMessage {

public static void main(String[] args) {

//1) open the browser

ChromeDriver driver = new ChromeDriver();

//2)Navigate to application

driver.get("https://facebook.com");

// 3) Enter invalid username 'batman554466@gmail.com' in the Email address or phone number' textbox

driver.findElement(By.id("email")).sendKeys("batman554466@gmail.com");

//4)Enter invalid password 'password@123' in the 'Password' textbox

driver.findElement(By.id("pass")).sendKeys("password@123");

//5) Click Login button

driver.findElement(By.name("login")).click();

//6) Verify user sees the error message- "The email address you entered isn't connected to an account. Find your account and log in."

String expectedErrMsg = "The email address you entered isn't connected to an account. Find your account and log in.";

String actualErrMsg = driver.findElement(By.xpath("//\*[@id=\"email\_container\"]/div[2]")).getText();

if(expectedErrMsg.equals(actualErrMsg)) {

System.out.println("Test case Passed");

}

else {

System.out.println("Test case Failed");

}

driver.quit();

//

}

}

5. Demonstrate how web elements are handled in Selenium.

6. Demonstrate how to automate calendars on the web page.

package demo;

import java.util.List;

import org.openqa.selenium.By;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class Calendar {

public static void main(String[] args) {

//1)Open the Browser

ChromeDriver driver = new ChromeDriver();

//2) Maximize it

driver.manage().window().maximize();

//3) Navigate to application

driver.get("https://www.expedia.co.in/");

//4)Click on Ca;endar icon=d1-btn

driver.findElement(By.id("date\_form\_field-btn")).click();

//5)Select 9tj from the next month

WebElement nextmonth = driver.findElement(By.xpath("(//table[contains(@class,'weeks')])[2]"));

List<WebElement> rows = nextmonth.findElements(By.tagName("tr"));

for(int i = 1;i<rows.size();i++) {

WebElement row = rows.get(i);

List<WebElement> columns = row.findElements(By.tagName("button"));

for(WebElement x : columns) {

if(x.getAttribute("data-day").equals("9")) {

x.click();

break;//stop searching rest of the dates by coming out of the for-each loop

}

}

}

//6)Click on 'Done' to close the calendar

driver.findElement(By.xpath("//button[contains(@data-stid,'apply-date-picker')]")).click();

}

}

7. Using Selenium WebDriver, write a program to handle alerts.

**package** demo;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** WindowHandleDemo {

**public** **static** **void** main(String[] args) {

ChromeDriver driver = **new** ChromeDriver();

System.***out***.println(driver.getWindowHandle());

}

}

}

package demo;

import org.openqa.selenium.By;

import org.openqa.selenium.chrome.ChromeDriver;

public class AlertBoxDemo {

public static void main(String[] args) throws InterruptedException {

//1) Open the Browser

ChromeDriver driver = new ChromeDriver();

//2) MAximize it

driver.manage().window().maximize();

//3)Navigate to application

driver.get("https://retail.onlinesbi.sbi/retail/login.htm");

//4)Click on 'CONTINUE TO LOGIN' button

driver.findElement(By.linkText("CONTINUE TO LOGIN")).click();

//5)Click on 'Login' Button

driver.findElement(By.id("Button2")).click();

Thread.sleep(3000);

//6)Close the Alert Box

driver.switchTo().alert().accept();

}

}

9. Demonstrate installation and configuration of AutoIT.

**Development Environment:**

* Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
* JavaDevelopment Kit Version 8

This guide has mainly two subsection, namely:

4.10.1 Installing and configuring Auto IT

4.10.2 Pushing the code to GitHub repositories

**Step 4.10.1:** Installing and Configuring Auto IT

* Download Auto IT from <https://www.autoitscript.com/site/autoit/downloads/> link.
* Save it in one folder.
* Double click on autoit-v3-setup.exe file and click on **Install**.
* After successful installation, open up AutoIT Editor.

C:\Program Files(x86)\AutoIt3\SciTE

**Step 4.10.2:** Pushing the code to GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files:

Cd <folder path>

* Initialize your repository using the following command:

Git init

* Add all the files to Git repository using following command:

Git add.

* Commit the changes using the following command:

Git commit -m “add the comment”

* Push the files to the folder you initially created using the following command:

Git push -u origin master

10. Demonstrate how file uploads are handled in AutoIT.

**package** demo;

**import** java.io.IOException;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.chrome.ChromeDriver;

**public** **class** Uploadfile {

**public** **static** **void** main(String[] args) **throws** IOException ,InterruptedException {

//1)Open the browser

ChromeDriver driver = **new** ChromeDriver();

//2)MAximize it

driver.manage().window().maximize();

//3)NAviagte to application

driver.get("https://www.remove.bg/");

//4)Click 'Select files to upload 'Button

driver.findElement(By.*xpath*("(//button[contains(@type,'button')])[3]")).click();

Thread.*sleep*(3000);

//5)HAndover the control to autoIT to select a file from the local machine ,click on open button

Runtime.*getRuntime*().~~exec~~("resources//demo.exe");

}

}