1.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.

- **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.WebDriver Installation and Integration in Eclipse

Development Environment

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

Step 1: Download Selenium Standalone Server jar

Step 2: Launching Eclipse and creating a Java project

- Launch the Eclipse and create a Workspace.
- Create Project:
- Click on File -> New -> Java Project.

Step 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.

3.Multiple ways to locate elements

Step 1: Using ID as a locator

- Open Eclipse
- Find a web element using Locator **ID**
- Syntax: id = id of the element
- Example: driver.findElement(By.id("Email"));

Step 2: Using class name as a locator

- Find aweb element using locator **ClassName**
    - Syntax: class = Class Name of the element
    - Example: driver.findElement(By.class("classname"));

Step 3: Using name as a locator

- Find aweb element using locator name
    - Syntax: name = Name of the element
    - Example: driver.findElement(By.name("name"));

Step 4: Using LinkText as a locator

- Find aweb element using Locator Link Text
    - Syntax: link = partialLink of the element
    - Example: driver.findElement(By.partialLinkText("plink"));

Step 5: Using Xpath as a locator

- Find aweb element using locator **Xpath**
- Xpath can be created in two ways:

    a. **Relative Xpath**
        - Syntax: relativeXpath : //*[@class='relativexapath']
        - Example: driver.findElement(By.xpath("//*[@class='relativexapath']"));

    b. **Absolute Xpath**
        - Syntax: absoluteXpath : html/body/div[1]/div[1]/div/h4[1]/b
        - Example: driver.findElement(By.xpath("html/body/div[1]/div[1]/div/h4[1]/b"));

Step 6: Using Xpath as a **CSS Selector**

- CSS Selector have many formats, namely:

- **Tag and ID**
  - Syntax:"css = tag#id"
  - Example: driver.findElement(By.cssSelector("input#email"));

- **Tag and Class**
  - Syntax: "css = tag.class"
  - Example: driver.findElement(By.cssSelector("input.inputtext"));

- **Tag and Attribute**
  - Syntax: "css = tag[attribute=value]"
  - Example: driver.findElement(By.cssSelector("input[name=lastName]"));

- **Tag, Class, and Attribute**
  - Syntax: "tag.class[attribute=value]"
  - Example: driver.findElement(By.cssSelector("input.inputtext[tabindex=1]"));

- **Inner text**
  - Syntax: "css = tag.contains("innertext")"
  - Example: driver.findElement(By.cssSelector(font:contains("Boston")));

Step 7: Using Xpath for handling complex and dynamic elements

- Dynamic Xpath has many formats, namely:

- **Contains();**
  - Syntax: "xpath = //*[contains(text(),'text')]
  - Example: driver.findElement(By.xpath("//*[contains(text(),'sub']"));

- **Using OR & AND**
  - Syntax: xpath=//*[@type='submit' or @name='btnReset']
  - Example:
  - driver.findElement (By.xpath("=//*[@type='submit' or @name='btnReset']"));

- **Start-with function**
  - Syntax: xpath= //label[starts-with(@id,'message')]
  - Example:
    - driver.findElement (By.xpath("//label[starts-with(@id,'message')]"));

- **Text();**
  - Syntax: xpath=//td[text()='UserID']
  - Example: : driver.findElement (By.xpath("=//td[text()='UserID']"));

- **Following**

- Syntax: xpath=//*[@type='text']//following::input
- Example: driver.findElement(By.xpath("=//*[@type='text']//following::input"));

- **Preceding**
  - Syntax: xpath=//*[@type='text']//preceding::input
  - Example: driver.findElement(By.xpath("//*[@type='text']//preceding::input"));

- **Following - sibling**
  - Syntax: xpath=//*[@type='submit']//preceding::input
  - Example:
  - driver.findElement (By.xpath ("//*[@type='text']//following-sibling::input"));

- The code for the above steps is as follows:

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;


public class LocatorExample {

   public static void main(String[] args) {

      // Set the path to the ChromeDriver executable

System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");


// Create a new instance of the ChromeDriver

      WebDriver driver = new ChromeDriver();


      // Launch the website

driver.get("https://www.seleniumhq.org");


      // Step 1: Using ID as a Locator

WebElementdownloadTab = driver.findElement(By.id("menu_download"));

downloadTab.click();


      // Step 2: Using class name as a Locator
```

```
WebElementversionsTab = driver.findElement(By.className("version"));
versionsTab.click();

    // Step 3: Using Name as a Locator
WebElementsearchField = driver.findElement(By.name("q"));
searchField.sendKeys("Selenium WebDriver");

    // Step 4: Using LinkText as a Locator
WebElementaboutTab = driver.findElement(By.linkText("About"));
aboutTab.click();

    // Step 5: Using Xpath as a Locator
    // Relative Xpath
WebElementrelativeElement = driver.findElement(By.xpath("//a[@class='navbar-brand']"));
relativeElement.click();

    // Absolute Xpath
WebElementabsoluteElement = driver.findElement(By.xpath("/html/body/div[1]/div[1]/a"));
absoluteElement.click();

    // Step 6: Using CSS Selector as a Locator
    // Tag and ID
WebElementdownloadButton = driver.findElement(By.cssSelector("a#downloadSeleniumBtn"));
downloadButton.click();

    // Tag and Class
WebElementprojectName = driver.findElement(By.cssSelector("h1.project-name"));
System.out.println("Project Name: " + projectName.getText());

    // Step 7: Using Xpath for handling complex and dynamic elements
    // Contains()
```

```
WebElementprojectsLink = driver.findElement(By.xpath("//*[contains(text(),'Projects')]"));
projectsLink.click();

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

4.Locating elements through CSS and XPath

Step 1: Finding the element present on the page using CSS Selector.

- Open Eclipse
- Using Path as a CSS Selector
- CSS Selector has many formats, namely:
    a. **Tag and ID**
        - Syntax: “css = tag#id”
        - Example: driver.findElement(By.cssSelector(“input#email”));

    b. **Tag and Class**
        - Syntax: “css = tag.class”
        - Example: driver.findElement(By.cssSelector(”input.inputtext”));

    C. **Tag and Attribute**
        - Syntax: “css = tag[attribute=value]”
        - Example: driver.findElement(By.cssSelector(“input[name=lastName]”));

    c. **Tag, Class, and Attribute**
        - Syntax: “tag.class[attribute=value]”
        - Example:
          driver. findElement(By.cssSelector(“input.inputtext[tabindex=1]”));

    d. **Inner text**

- Syntax: “css = tag.contains(“innertext”)”
- Example: driver.findElement(By.cssSelector(font:contains(“Boston”)));

Step 2: Finding the element present on the page using Path.

- In Selenium automation, if the elements are not found by the general locators like id, class, name, etc., then XPath is used to find an element on the web page.
- XPath contains the path of the element situated at the web page. Standard syntax for creating XPath is:

  XPath=//tagname[@attribute='value']

  - **//:** Select current node.
  - **Tagname:** Tagname of the particular node.
  - **@:** Select attribute.
  - **Attribute:** Attribute name of the node.
  - **Value:** Value of the attribute.
- Types of XPath:

There are two types of XPath:

a. **Absolute XPath**
   - It is a direct way to find the element, but the disadvantage of the absolute XPath is that if there are any changes made in the path of the element, then that XPath fails.
   - The key characteristic of XPath is that it begins with the single forward slash (/), which means you can select the element from the root node.
   - Syntax for absolute Path: html/body/div[1]/div[1]/div/h4[1]/b
   - Example: driver.findElement(By.xpath(“html/body/div[1]/div[1]/div/h4[1]/b”));
   - Writing absolute XPath on the elements which are present in the web page will be very lengthy. To reduce the length, we use relative XPath.

b. **Relative XPath**
   - For relative XPath, the path starts from the middle of the HTML DOM structure. It starts with the double forward-slash (//), which means it can search the element anywhere on the web page.
   - You can start from the middle of the HTML DOM structure and you don’t need to write long XPath.
   - Syntax for relativeXPath: //*[@class=’relativexapath’]
   - Example: driver.findElement(By.xpath(“//*[@class=’relativexapath’]”))

The code for the above is as follows:

```
import org.openqa.selenium.By;

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

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

public class ElementLocatorExample {
    public static void main(String[] args) {
        // Set the path to the ChromeDriver executable
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        // Launch the Chrome browser
        WebDriver driver = new ChromeDriver();

        // Navigate to the W3Schools HTML Examples page
driver.get("https://www.w3schools.com/html/html_examples.asp");

        // Locating elements using CSS selectors
        // example 1: Tag and ID
WebElement element1 = driver.findElement(By.cssSelector("a#topnavbtn_references"));

        // example 2: Tag and Class
WebElement element2 = driver.findElement(By.cssSelector("h2.w3-container.w3-red"));

        // example 3: Tag and Attribute
WebElement element3 = driver.findElement(By.cssSelector("img[alt='W3Schools.com']"));

        // example 4: Tag, Class, and Attribute
WebElement element4 = driver.findElement(By.cssSelector("div.w3-panel.w3-leftbar.w3-sand.w3-
padding"));

        // example 5: Inner text
WebElement element5 = driver.findElement(By.cssSelector("a:contains('W3Schools')"));

        // Locating elements using XPath
```

```
        // Example 1: Absolute XPath
WebElement element6 = driver.findElement(By.xpath("/html/body/div[5]/div[1]/div[1]/div[4]/h2"));


        // example 2: Relative XPath
WebElement element7 = driver.findElement(By.xpath("//*[@class='w3-sidebar w3-bar-block w3-light-grey w3-card']//a[contains(text(),'Try it Yourself')]"));


        // Perform actions on the located elements
        // ...


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

5.Handling various web elements

Step 1: Edit box

- Open Eclipse
- It is a basic text control that enables a user to type a small amount of text.
- Operations on Edit box
    - Enter a Value,
    - Clear the Value,
    - Check enabled status,
    - Check edit box existence,
    - Get the value

Step 2: Link

- Link is more appropriately referred to as a hyperlink and connects one web page to another. It allows the user to click their way from page to page.
- Operations on Link
    - Click the link,
    - Check the link existence,
    - Check the link enabled status,
    - Return the link name

Step 3: Button

- This represents a clickable button, which can be used in forms and places in the document that needs a simple, standard button functionality.
- Operations on Button
    - Click
    - Check Enabled status
    - Display status

Step 4: Image, image link, and image button

- It helps in performing actions on images like clicking on the image link or the image button, etc.
- Operations Image
    - Three types of Image elements in Web Environment
    - General Image (No functionality)
    - Image Button (Submits)
    - Image Link (Redirects to another page/location)

Step 5: Text area

- It is an inline element used to designate a plain-text editing control containing multiple lines.

- Return / Capture Text Area or Error message from a web page

Step 6: Checkbox

- This is a selection box or a tick box which is a small interactive box that can be toggled by the user to indicate an affirmative or a negative choice.
- Operations on checkbox
    - Check if the checkbox is displayed or not
    - Check if the checkbox is enabled or not
    - Check if the checkbox is selected or not
    - Select the checkbox
    - Unselect the checkbox

Step 7: Radio button

- It is an option button which is a graphical control element that allows the user to choose only one predefined set of mutually exclusive options.
- Operations on Radio Button
    - Select Radio Button
    - Verify if the Radio Button is displayed or not
    - Verify if the Radio Button is enabled or not
    - Verify if the Radio Button is selected or not

Step 8: Drop-down list

- It is a graphical control element, similar to the list box, which allows the user to choose one value from the list. When this drop-down list is inactive, it displaysonly a single value.
- Operations on drop-down list
    - Check the drop-down box's existence
    - Check if the drop-down is enabled or not
    - Select an item
    - Get Items Count

Step 9: Web table/HTML table

- Operations on Web Table/HTML Table
    - Get cell value
    - Get Rows Count
    - Get Cells Count

Step 10: Frame

- Operations on Frame
    - Switch from Top window to a frame
    - Switch from a frame to Top window

Step 11: Switching between tabs in the same browser window

- Open a new tab using Ctrl + t.
- Driver control automatically switches to the newly opened tab.
- Perform the required operations here.
- Next, switch back to the old tab using Ctrl + Tab. You need to keeppressing this unless you reach the desired tab.
- Once the desired tab is reached, then perform the operations in thattab.

Here is the code for the above steps:

```
import org.openqa.selenium.By;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.Select;


import java.util.ArrayList;


public class WebElementsDemo {
```

```
    public static void main(String[] args) {
        // Set the path to the chromedriver executable
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

        // Launch the Chrome browser
        WebDriver driver = new ChromeDriver();

        // Navigate to Google homepage
driver.get("https://www.google.com");



        // Step 1: Edit box
        // Enter a value in the search box
driver.findElement(By.name("q")).sendKeys("Hello World");

        // Clear the value
driver.findElement(By.name("q")).clear();

        // Check enabled status of the search box
booleanisSearchBoxEnabled = driver.findElement(By.name("q")).isEnabled();

        // Check search box existence
booleandoesSearchBoxExist = driver.findElements(By.name("q")).size() > 0;

        // Get the value from the search box
        String searchBoxValue = driver.findElement(By.name("q")).getAttribute("value");

        // Step 2: Link
        // Click the "Images" link
driver.findElement(By.linkText("Images")).click();
```

```
        // Check the "Images" link existence
booleandoesImagesLinkExist = driver.findElements(By.linkText("Images")).size() > 0;

        // Check the "Images" link enabled status
booleanisImagesLinkEnabled = driver.findElement(By.linkText("Images")).isEnabled();

        // Return the "Images" link name
        String imagesLinkName = driver.findElement(By.linkText("Images")).getText();

        // Step 3: Button
        // Click the "Google Search" button
driver.findElement(By.name("btnK")).click();

        // Check enabled status of the "Google Search" button
booleanisSearchButtonEnabled = driver.findElement(By.name("btnK")).isEnabled();

        // Display status of the "Google Search" button
booleanisSearchButtonDisplayed = driver.findElement(By.name("btnK")).isDisplayed();

        // Step 4: Image, image link, and image button
        // Operations on Image

        // Step 5: Text area
        // Return / Capture Text Area or Error message from a web page
        String textAreaValue = driver.findElement(By.tagName("textarea")).getAttribute("value");

        // Step 6: Checkbox
        // Check if the "I'm Feeling Lucky" checkbox is displayed or not
booleanisFeelingLuckyCheckboxDisplayed = driver.findElement(By.name("btnI")).isDisplayed();

        // Check if the "I'm Feeling Lucky" checkbox is enabled or not
```

```
booleanisFeelingLuckyCheckboxEnabled = driver.findElement(By.name("btnI")).isEnabled();

    // Check if the "I'm Feeling Lucky" checkbox is selected or not
booleanisFeelingLuckyCheckboxSelected = driver.findElement(By.name("btnI")).isSelected();

    // Select the "I'm Feeling Lucky" checkbox
driver.findElement(By.name("btnI")).click();

    // Unselect the "I'm Feeling Lucky" checkbox
driver.findElement(By.name("btnI")).click();

    // Step 7: Radio button
    // Select the "Search" radio button
driver.findElement(By.cssSelector("input[name='tbm'][value='s']")).click();

    // Verify if the "Search" radio button is displayed or not
booleanisSearchRadioButtonDisplayed =
driver.findElement(By.cssSelector("input[name='tbm'][value='s']")).isDisplayed();

    // Verify if the "Search" radio button is enabled or not
booleanisSearchRadioButtonEnabled =
driver.findElement(By.cssSelector("input[name='tbm'][value='s']")).isEnabled();

    // Verify if the "Search" radio button is selected or not
booleanisSearchRadioButtonSelected =
driver.findElement(By.cssSelector("input[name='tbm'][value='s']")).isSelected();

    // Step 8: Drop-down list
    // Check the drop-down box existence
booleandoesDropDownExist = driver.findElements(By.name("lang")).size() > 0;

    // Check if the drop-down box is enabled or not
```

```
booleanisDropDownEnabled = driver.findElement(By.name("lang")).isEnabled();

        // Select an item from the drop-down list
        Select dropdown = new Select(driver.findElement(By.name("lang")));
dropdown.selectByVisibleText("English");
dropdown.selectByValue("fr");

        // Get the count of items in the drop-down list
        int itemCount = dropdown.getOptions().size();

        // Step 9: Web table/HTML table
        // Get cell value from a table
        String cellValue = driver.findElement(By.xpath("//table/tbody/tr[1]/td[1]")).getText();

        // Get the count of rows in a table
        int rowCount = driver.findElements(By.xpath("//table/tbody/tr")).size();

        // Get the count of cells in a table row
        int cellCount = driver.findElements(By.xpath("//table/tbody/tr[1]/td")).size();

        // Step 10: Frame
        // Switch from the default content to a frame
driver.switchTo().frame("frameName");

        // Switch back from a frame to the default content
driver.switchTo().defaultContent();

        // Step 11: Switching between tabs in the same browser window
        // Open a new tab
driver.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL + "t");
```

```
        // Switch to the newly opened tab
ArrayList<String> tabs = new ArrayList<>(driver.getWindowHandles());
driver.switchTo().window(tabs.get(1));

        // Perform operations in the new tab

        // Switch back to the old tab
driver.findElement(By.cssSelector("body")).sendKeys(Keys.CONTROL + "\t");
driver.switchTo().defaultContent();

        // Perform operations in the old tab

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

6.Automate Calendars on web page

Step 1: Create a Selenium project

- Open Eclipse and create a new Java project.
- Add selenium jar files to the build path.
- Add browser executable files in the resource folder.

Step 2: Write code for calendar automation

- Create a Java file with the name calendar.java and write the code given below:

```
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class DatePicker
{
```

```
public static void main(String[] args) throws InterruptedException
{
	String dot="11/December/2023";
	String date,month,year;
	String caldt,calmonth,calyear;
	/*
	 * Split the String into String Array
	 */
	String dateArray[]= dot.split("/");
	date=dateArray[0];
	month=dateArray[1];
	year=dateArray[2];

	ChromeDriver driver=new ChromeDriver();
	driver.get("http://cleartrip.com");
	driver.findElement(By.id("DepartDate")).click();

	WebElementcal;
	cal=driver.findElement(By.className("calendar"));
	calyear=driver.findElement(By.className("ui-datepicker-year")).getText();
	/**
	 * Select the year
	 */
	while (!calyear.equals(year))
	{
		driver.findElement(By.className("nextMonth")).click();
		calyear=driver.findElement(By.className("ui-datepicker-year")).getText();
		System.out.println("Displayed Year::" + calyear);
	}

	calmonth=driver.findElement(By.className("ui-datepicker-month")).getText();
```

```
		/**
		 * Select the Month
		 */
		while (!calmonth.equalsIgnoreCase(month))
		{
			driver.findElement(By.className("nextMonth ")).click();
			calmonth=driver.findElement(By.className("ui-datepicker-
month")).getText();
		}

		cal=driver.findElement(By.className("calendar"));
		/**
		 * Select the Date
		 */
		List<WebElement>rows,cols;
		rows=cal.findElements(By.tagName("tr"));
		for (int i = 1; i<rows.size(); i++)
		{
			cols=rows.get(i).findElements(By.tagName("td"));
			for (int j = 0; j <cols.size(); j++)
			{
				caldt=cols.get(j).getText();
				if (caldt.equals(date))
				{
					cols.get(j).click();
					break;
				}
			}
		}

	}
```

}

- Run the project as a Java application.

7.program to handle Alerts

```
package com.seleniumtest.exwebelement;

import java.time.Duration;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDrive;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

/**
 * This class demonstrate JavaScript Alerts Automation Test.
 *
 */
public class JsAlertsTest {

	static WebDriver driver;

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

		setUp();

		// test alert
		testAlert();
```

```
		// test confirmation alert
		testConfirmDailogAlert1();
		testConfirmDailogAlert2();

		// test prompt alert accept
		testPromptDailogAlert1();
		testPromptDailogAlert2();
	}

	public static void setUp() {

		// step1: formulate a test domain url & driver path
		String siteUrl = "file:///C:\\Users\\\\eclipse-workspace\\Phase1-JDBC-Selenium-Test-06-12-2023\\static\\alert-web-elements.html";
		String driverPath = "drivers/windows/geckodriver.exe";

		// step2: set system properties for selenium dirver
		System.setProperty("webdriver.chrome.driver", driverPath);

		// step3: instantiate selenium webdriver
		driver = new FirefoxDriver();

		// step4: add implicit wait (Unconditional Delay)
		driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(50));

		// step5: launch browser
		driver.get(siteUrl);
	}

	public static void testAlert() throws InterruptedException {
		// find alert button and click
```

```
		driver.findElement(By.cssSelector("body > p:nth-child(2) > button:nth-
child(1)")).click();

		// switch to sub alert window
		Alert alert = driver.switchTo().alert();

		String expText = "Do you really want to proceed ??";
		if (expText.equals(alert.getText())) {
			System.out.println("Test is passed !");
		} else {
			System.out.println("Test is failed !");
		}

		Thread.sleep(2000);

		// accept the alert
		alert.accept();
	}

	// Test confirmation alert and accept
	public static void testConfirmDailogAlert1() throws InterruptedException {
		// find alert button and click
		driver.findElement(By.cssSelector("body > p:nth-child(4) > button:nth-
child(1)")).click();

		// switch to sub confirmation alert window
		Alert alert = driver.switchTo().alert();

		String expText = "Please confirm the action is right ??";
		if (expText.equals(alert.getText())) {
			System.out.println("Test is passed !");
		} else {
```

```
                System.out.println("Test is failed !");
            }

            Thread.sleep(2000);

            // accept the alert
            alert.accept();
        }

        // Test confirmation alert and cancel
        public static void testConfirmDailogAlert2() throws InterruptedException {
            // find alert button and click
            driver.findElement(By.cssSelector("body > p:nth-child(4) > button:nth-
child(1)")).click();

            // switch to sub confirmation alert window
            Alert alert = driver.switchTo().alert();

            String expText = "Please confirm the action is right ??";
            if (expText.equals(alert.getText())) {
                System.out.println("Test is passed !");
            } else {
                System.out.println("Test is failed !");
            }

            Thread.sleep(2000);

            // cancel the alert
            alert.dismiss();
        }

        // Test prompt alert and accpet
```

```
public static void testPromptDailogAlert1() throws InterruptedException {
	// find alert button and click
	driver.findElement(By.xpath("/html/body/p[6]/button")).click();

	// switch to sub confirmation alert window
	Alert propmt = driver.switchTo().alert();

	String expText = "Please enter your name.";
	if (expText.equals(propmt.getText())) {
		System.out.println("Test is passed !");
	} else {
		System.out.println("Test is failed !");
	}

	Thread.sleep(2000);

	// cancel the alert
	propmt.accept();
}

// Test prompt alert and cancel
public static void testPromptDailogAlert2() throws InterruptedException {
	// find alert button and click
	driver.findElement(By.xpath("/html/body/p[6]/button")).click();

	// switch to sub confirmation alert window
	Alert propmt = driver.switchTo().alert();

	String expText = "Please enter your name.";

	propmt.sendKeys("John Smith");
```

```
            if (expText.equals(propmt.getText())) {
                System.out.println("Test is passed !");
            } else {
                System.out.println("Test is failed !");
            }

            Thread.sleep(2000);

            // cancel the alert
            propmt.dismiss();
        }

}
```

8.Screenshots and browser profiles

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- JavaDevelopment Kit Version 8
- Selenium standalone server Version 3.141.59

Step 1: Screenshots

- Open Eclipse
- Convert web driver object to **TakeScreenshot**
- Call getScreenshotAs method to create image file
- Copy file to desired location

Step 2: Convert web driver object to TakeScreenshot

- Syntax: TakesScreenshotscrShot = (TakesScreenshot)driver;

Step 3: Call getScreenshotAs method to create image file

- Syntax: File srcFile = scrShot.getScreenshotAs(OutType.FILE);

Step 4: Copy file to desire location

- Syntax: FileUtils.copyFile(source, filePath);
- The script looks like this:

```
package screenshots.screenshot;
import java.io.File;
import java.io.IOException;
import org.openqa.selenium.By;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import com.sun.jna.platform.FileUtils;


public class Screenshots {
 public static void main(String[] args ) throws IOException
  {
        System.setProperty("webdriver.chrome.driver",
"C:\\Users\\VICKY\\chromedriver_win32\\chromedriver.exe");
    WebDriver driver = new ChromeDriver();
  driver.get("https://www.flipkart.com/");
  WebElement upload =  driver.findElement(By.xpath("//*[@type='text']"));
  upload.click();
  TakesScreenshotts = (TakesScreenshot)driver;
    File scr = ts.getScreenshotAs(OutputType.FILE);
  FileUtils.copyFile(scr, new File("/Screenshot/test.png");
```

```
        }
    }
```

9.Auto IT Installation and Configuration

Development Environment:

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

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

10.Handling file uploads in AutoIT

Development Environment

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- JavaDevelopment Kit Version 8
- Selenium standalone server
- AutoIT

Step 1: Handling file upload by SendKeys

- Launch Eclipse and create a Java project.
- Create project: Click on file->New->Java project.
- Enter the project name as UploadFile and click on Finish.
- In the project explorer,expand UploadFile.
- Right-click on src and choose New->Class.

- In Package Name, enter com.ecommerceand in Name enter Upload and click on Finish.
- Locate the browse button using chropath/firebug.
- Set the path using SendKeys. And the code looks like below:

```
//Locating 'browse' button
WebElement browse =driver.findElement(By.id("uploadfile"));
//pass the path of the file to be uploaded using Sendkeys method
browse.sendKeys("D:\\SoftwareTestingMaterial\\UploadFile.txt");
```

Step 2: Handling file upload by AutoIT script

- Go to Start->Autoit v3->Autoit window info.
- Now drag the Finder toolbox to the object in which you are interested.

- Build an AutoIT script using SciTE editor and write the script using ControlFocus, ControlsetText, and ControlClick commands.
- And the script looks like below:
- Save the Script with .au3 extension.
- Compile the .au3 script which converts into .exe file.
- Pass the .exepath into selenium test script using method
- Runtime.getRuntime().exec("C:\AutoIt\Autoitscript.exe")
- The complete script looks like this:

```
import java.io.IOException;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class AutoIt {
private static WebDriver driver = null;
public static void main(String[] args) throws IOException, InterruptedException {
driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://toolsqa.com/automation-practice-form");
driver.findElement(By.id("photo")).click();
```

```
Runtime.getRuntime().exec("D:\AutoIt\AutoItTest.exe");

Thread.sleep(5000);

driver.close();
```

11.Sikuli for UI Testing

Development Environment:

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

Step 1: Integrating Sikuli with Selenium WebDriver

- Open Eclipse and create a new Java project
- Right-click on the project. Navigate through the given path: Build path->Configure build path->Add external Jars.
- Click on **Apply and OK.**

Step 2: Screen class in Sikuli

- Screen class is a base class which contains some predefined methods to perform operations, such as click, double click, providing input to the text box and hover, etc.

- Below are the commonly used methods:

    - Click

Syntax: Screen s = new Screen();
s.click()("imag.png”);

    - doubleClick

Syntax: Screen s = new Screen();
s.doubleClick()("imag.png”);

- Type

Syntax: s.type(“imag.png”, “Text”);

- Hover

Syntax: s.hover(“imag.png”);

- Find

Syntax: s.find(“imag.png”);

Step 3: Pattern class in Sikuli

- Pattern class is used to associate the image file to identify the element
- Pattern class takes the path of the image as a parameter
- Below are the commonly used methods:

  - getFileName

  Syntax: Pattern p = new Pattern(“D:\Test\imag.png”)

  - Similar

  Syntax: Pattern p1 = p.similar Pattern(“0.7f”);

  - Exact

  Syntax: Pattern p1 = p.exact();

The script looks like this:

```
package sikuli1;

import org.sikuli.script.FindFailed;

import org.sikuli.script.Pattern;

import org.sikuli.script.Screen;


public class SikuliClass {

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

		Screen s = new Screen();

		Pattern p = new
Pattern("C:\\Users\\Testing\\Desktop\\siluli\\Capture.PNG");

		s.doubleClick(p);

	}
```

12. JDBC in Selenium

Development Environment:

- Eclipse IDE for Enterprise Java Developers Version Oxygen.3a Release (4.7.3a)
- JavaDevelopment Kit Version 8
- Selenium Standalone Server Version 3.141.59
- Structured query language server Version SQL Server 2016 SP2

Step 1: Creating a table in Database

- Create a table and enter the data in the table in the Database.

Step 2: Writing the JDBC connection integrating with selenium

- Load the driver class
  Syntax: class.forName("Connection URL");

- Create a Connection
  Connection con = DriverManager.getConnection("URL", "UserName", "Password");

- Create a statement
  Syntax: Statement stmt = con.createStatement();

- Execute SQL query
  Syntax: ResultSetrs= stmt.executeQuery("sql query");

- Close the connection
  Syntax: Con.close();

The code in Eclipse will look like this:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;
```

```
import java.sql.Statement;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class TestDatabaseWithSelenium {
   private WebDriver driver;

   @BeforeTest
   public void setup() {
      // Set the path to the ChromeDriver executable
System.setProperty("webdriver.chrome.driver", "path/to/chromedriver");

      // Create a new instance of the ChromeDriver
      driver = new ChromeDriver();
   }

   @Test
   public void testVerifyDB() throws ClassNotFoundException, SQLException {
      // Step 1: Load the driver class
Class.forName("oracle.jdbc.driver.OracleDriver");

      // Step 2: Create the connection object
      Connection con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "your_username",
"your_password");

      // Step 3: Create the statement object
```

```
        Statement stmt = con.createStatement();

        // Step 4: Execute the SQL query
ResultSetrs = stmt.executeQuery("SELECT * FROM Products");

        // Step 5: Iterate through the result set and perform web testing
        while (rs.next()) {
            int productId = rs.getInt(1);
            String productName = rs.getString(2);
            String productDescription = rs.getString(3);

            // Perform web testing using Selenium
driver.get("https://www.seleniumhq.org");
WebElementsearchInput = driver.findElement(By.id("q"));
searchInput.sendKeys(productName);
searchInput.submit();

            // Print the database record and web page title
System.out.println("Product ID: " + productId);
System.out.println("Product Name: " + productName);
System.out.println("Product Description: " + productDescription);
System.out.println("Web Page Title: " + driver.getTitle());
System.out.println("----------------------------------------");
        }

        // Step 6: Close the connection object
con.close();
    }

    @AfterTest
    public void teardown() {
```

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