| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Practical Number:** 4 | |
| **Aim of Practical:**  1.Demonstrate different types of alerts  2. Demonstrate :  Handling Drop Down  List Boxes  3. Demonstrate  Command Button  Radio buttons & text boxes  Waits command in selenium  4. Demonstrate action classes in Selenium | | | |
| **DOP:** 20.10.23 | | **DOS:** 1.11.23 | |
| **CO Mapped:** - | **PO Mapped:** - | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 4

**Aim:** -

1.Demonstrate different types of alerts

2. Demonstrate :

Handling Drop Down

List Boxes

3. Demonstrate

Command Button

Radio buttons & text boxes

Waits command in selenium

4. Demonstrate action classes in Selenium

**Theory:**

WebDriver provides an API for working with the three types of native popup messages offered by JavaScript. These popups are styled by the browser and offer limited customisation.

The simplest of these is referred to as an alert, which shows a custom message, and a single button which dismisses the alert, labelled in most browsers as OK. It can also be dismissed in most browsers by pressing the close button, but this will always do the same thing as the OK button.

Perhaps the most common challenge for browser automation is ensuring that the web application is in a state to execute a particular Selenium command as desired. The processes often end up in a race condition where sometimes the browser gets into the right state first (things work as intended) and sometimes the Selenium code executes first (things do not work as intended). This is one of the primary causes of flaky tests.

All navigation commands wait for a specific readyState value based on the page load strategy (the default value to wait for is "complete") before the driver returns control to the code. The readyState only concerns itself with loading assets defined in the HTML, but loaded JavaScript assets often result in changes to the site, and elements that need to be interacted with may not yet be on the page when the code is ready to execute the next Selenium command.

Similarly, in a lot of single page applications, elements get dynamically added to a page or change visibility based on a click. An element must be both present and displayed on the page in order for Selenium to interact with it.

Take this page for example: https://www.selenium.dev/selenium/web/dynamic.html When the “Add a box!” button is clicked, a “div” element that does not exist is created. When the “Reveal a new input” button is clicked, a hidden text field element is displayed. In both cases the transition takes a couple seconds. If the Selenium code is to click one of these buttons and interact with the resulting element, it will do so before that element is ready and fail.

The first solution many people turn to is adding a sleep statement to pause the code execution for a set period of time. Because the code can’t know exactly how long it needs to wait, this can fail when it doesn’t sleep long enough. Alternately, if the value is set too high and a sleep statement is added in every place it is needed, the duration of the session can become prohibitive.

Selenium provides two different mechanisms for synchronization that are better.

Implicit waits

Selenium has a built-in way to automatically wait for elements called an implicit wait. An implicit wait value can be set either with the timeouts capability in the browser options, or with a driver method (as shown below).

This is a global setting that applies to every element location call for the entire session. The default value is 0, which means that if the element is not found, it will immediately return an error. If an implicit wait is set, the driver will wait for the duration of the provided value before returning the error. Note that as soon as the element is located, the driver will return the element reference and the code will continue executing, so a larger implicit wait value won’t necessarily increase the duration of the session.

Warning: Do not mix implicit and explicit waits. Doing so can cause unpredictable wait times. For example, setting an implicit wait of 10 seconds and an explicit wait of 15 seconds could cause a timeout to occur after 20 seconds.

Explicit waits are loops added to the code that poll the application for a specific condition to evaluate as true before it exits the loop and continues to the next command in the code. If the condition is not met before a designated timeout value, the code will give a timeout error. Since there are many ways for the application not to be in the desired state, so explicit waits are a great choice to specify the exact condition to wait for in each place it is needed. Another nice feature is that, by default, the Selenium Wait class automatically waits for the designated element to exist.

Customization(Fluent):

The Wait class can be instantiated with various parameters that will change how the conditions are evaluated.

This can include:

Changing how often the code is evaluated (polling interval)

Specifying which exceptions should be handled automatically

Changing the total timeout length

Customizing the timeout message

For instance, if the element not interactable error is retried by default, then we can add an action on a method inside the code getting executed (we just need to make sure that the code returns true when it is successful).

**Code:**

File: Demo.java

package vesit.ajayk57.stqa.stqa\_practical4;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.time.Duration;

import java.util.Scanner;

import org.openqa.selenium.Alert;

import org.openqa.selenium.By;

import org.openqa.selenium.ElementNotInteractableException;

import org.openqa.selenium.JavascriptExecutor;

import org.openqa.selenium.Keys;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

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

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

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

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

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

public class Demo {

private final WebDriver driver;

private final Scanner sc = new Scanner(System.in);

public Demo(WebDriver driver) {

this.driver = driver;

}

public void runDifferentAlertsDemo() {

final String htmlDocumentName = "index.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

driver.get(htmlDocPath.toUri().toString());

runAlertBoxDemo();

runConfirmBoxDemo();

runPromptBoxDemo();

}

private void runAlertBoxDemo() {

Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(2));

Alert alert = wait.until(ExpectedConditions.alertIsPresent());

waitForEnterKey("Alert Box");

alert.accept();

}

private void runConfirmBoxDemo() {

WebElement button = driver.findElement(By.id("set\_confirmation\_button"));

button.click();

Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(2));

Alert alert = wait.until(ExpectedConditions.alertIsPresent());

waitForEnterKey("Confirm Box");

alert.dismiss(); // select Cancel

}

private void runPromptBoxDemo() {

WebElement button = driver.findElement(By.id("set\_name\_button"));

button.click();

Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(2));

Alert alert = wait.until(ExpectedConditions.alertIsPresent());

final String inputForPrompt = "Ajay Karthikesan";

alert.sendKeys(inputForPrompt);

alert.accept();

waitForEnterKey("Prompt Box");

}

public void runDropdownDemo() {

final String htmlDocumentName = "index2.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

driver.get(htmlDocPath.toUri().toString());

Select dropdown = new Select(driver.findElement(By.id("myDropdown")));

final String optionToSelect = "India";

dropdown.selectByVisibleText(optionToSelect);

waitForEnterKey("Dropdown");

}

public void runCommandButtonDemo() {

final String htmlDocumentName = "index3.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

driver.get(htmlDocPath.toUri().toString());

WebElement cmdButton = driver.findElement(By.cssSelector("button"));

waitForEnterKey("Command Button(Before Clicking]");

cmdButton.click();

waitForEnterKey("Command Button(After Clicking]");

}

public void runRadioButtonDemo() {

final String htmlDocumentName = "index3.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

driver.get(htmlDocPath.toUri().toString());

final String valueToSelect = "Java";

WebElement radioButton = driver

.findElement(By.cssSelector(String.format("input[type='radio'][value='%s']", valueToSelect)));

radioButton.click();

waitForEnterKey("Radio Button");

}

public void runTextBoxDemo() {

final String htmlDocumentName = "index3.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

driver.get(htmlDocPath.toUri().toString());

final String name = "Ajay Karthikesan";

WebElement textBox = driver.findElement(By.id("my\_name\_input"));

textBox.sendKeys(name);

WebElement submitButton = driver.findElement(By.cssSelector("input[type='submit']"));

submitButton.click();

waitForEnterKey("Textbox");

}

public void runWaitCommandsDemo() {

runImplicitWaitDemo();

runExplicitWaitDemo();

runFluentWaitDemo();

}

private void runImplicitWaitDemo() {

final String htmlDocumentName = "index3.html";

Path htmlDocPath = Paths.get(htmlDocumentName);

System.out.println("Waiting for 3 seconds(implicit wait)");

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(3));

driver.get(htmlDocPath.toUri().toString());

System.out.println("Current URL = " + driver.getCurrentUrl());

}

private void runExplicitWaitDemo() {

driver.get("https://www.selenium.dev/selenium/web/dynamic.html");

WebElement revealed = driver.findElement(By.id("revealed"));

Wait<WebDriver> wait = new WebDriverWait(driver, Duration.ofSeconds(2));

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

wait.until(d -> revealed.isDisplayed());

revealed.sendKeys("Displayed");

System.out.println("[ExplicitWait]Value of property 'value' = " + revealed.getDomProperty("value"));

}

private void runFluentWaitDemo() {

driver.get("https://www.selenium.dev/selenium/web/dynamic.html");

WebElement revealed = driver.findElement(By.id("revealed"));

Wait<WebDriver> wait = new FluentWait<>(driver).withTimeout(Duration.ofSeconds(2))

.pollingEvery(Duration.ofMillis(300)).ignoring(ElementNotInteractableException.class);

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

wait.until(d -> {

revealed.sendKeys("Displayed");

return true;

});

System.out.println("[FluentWait]Value of property 'value' = " + revealed.getDomProperty("value"));

}

public void runActionsDemo() {

runKeyboardActionsDemo();

runMouseActionsDemo();

wheelActionsDemo();

}

private void runKeyboardActionsDemo() {

driver.get("https://www.selenium.dev/selenium/web/single\_text\_input.html");

new Actions(driver).keyDown(Keys.SHIFT).sendKeys("a").keyUp(Keys.SHIFT).sendKeys("b").perform();

WebElement textField = driver.findElement(By.id("textInput"));

final String expectedValue = "Ab";

System.out.println("Expected Value: " + expectedValue);

System.out.println("Actual Value: " + textField.getAttribute("value"));

waitForEnterKey("Keyboard Actions");

}

private void runMouseActionsDemo() {

driver.get("https://www.selenium.dev/selenium/web/mouse\_interaction.html");

WebElement clickable = driver.findElement(By.id("clickable"));

new Actions(driver).doubleClick(clickable).perform();

final String expectedValue = "double-clicked";

System.out.println("Expected Value: " + expectedValue);

System.out.println("Actual Value: " + driver.findElement(By.id("click-status")).getText());

waitForEnterKey("Mouse Actions");

}

private void wheelActionsDemo() {

// scroll to element

driver.get(

"https://www.selenium.dev/selenium/web/scrolling\_tests/frame\_with\_nested\_scrolling\_frame\_out\_of\_view.html");

WebElement iframe = driver.findElement(By.tagName("iframe"));

new Actions(driver).scrollToElement(iframe).perform();

System.out.println("In Viewport = " + inViewport(iframe));

waitForEnterKey("Wheel Actions");

}

private boolean inViewport(WebElement element) {

String script = "for(var e=arguments[0],f=e.offsetTop,t=e.offsetLeft,o=e.offsetWidth,n=e.offsetHeight;\n"

+ "e.offsetParent;)f+=(e=e.offsetParent).offsetTop,t+=e.offsetLeft;\n"

+ "return f<window.pageYOffset+window.innerHeight&&t<window.pageXOffset+window.innerWidth&&f+n>\n"

+ "window.pageYOffset&&t+o>window.pageXOffset";

return (boolean) ((JavascriptExecutor) driver).executeScript(script, element);

}

private void waitForEnterKey(String demoName) {

while (true) {

System.out.println("Press enter after taking screenshots of " + demoName + " demo.");

String entered = sc.nextLine();

if (entered.equals("")) {

break;

}

}

}

public void end() {

sc.close();

driver.quit();

}

}

File: App.java

package vesit.ajayk57.stqa.stqa\_practical4;

import org.openqa.selenium.chrome.ChromeDriver;

/\*\*

\* Hello world!

\*

\*/

public class App {

public static void main(String[] args) {

Demo demo = new Demo(new ChromeDriver());

demo.runDifferentAlertsDemo();

demo.runDropdownDemo();

demo.runCommandButtonDemo();

demo.runRadioButtonDemo();

demo.runTextBoxDemo();

demo.runWaitCommandsDemo();

demo.runActionsDemo();

demo.end();

}

}

File: test:PenActionsTest.java

package vesit.ajayk57.stqa.stqa\_practical4;

import java.util.Arrays;

import java.util.List;

import java.util.Map;

import java.util.Scanner;

import java.util.stream.Collectors;

import org.junit.Test;

import org.junit.jupiter.api.Assertions;

import org.openqa.selenium.By;

import org.openqa.selenium.Rectangle;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.interactions.PointerInput;

public class PenActionsTest {

final private WebDriver driver = new ChromeDriver();

private final Scanner sc = new Scanner(System.in);

@Test

public void usePen() {

if (!(driver instanceof ChromeDriver)) {

System.err.println("ChromeDriver instance is needed for pen actions demo");

return;

}

driver.get("https://www.selenium.dev/selenium/web/pointerActionsPage.html");

WebElement pointerArea = driver.findElement(By.id("pointerArea"));

new Actions(driver).setActivePointer(PointerInput.Kind.PEN, "default pen").moveToElement(pointerArea)

.clickAndHold().moveByOffset(2, 2).release().perform();

List<WebElement> moves = driver.findElements(By.className("pointermove"));

Map<String, String> moveTo = getPropertyInfo(moves.get(0));

Map<String, String> down = getPropertyInfo(driver.findElement(By.className("pointerdown")));

Map<String, String> moveBy = getPropertyInfo(moves.get(1));

Map<String, String> up = getPropertyInfo(driver.findElement(By.className("pointerup")));

Rectangle rect = pointerArea.getRect();

int centerX = (int) Math.floor(rect.width / 2 + rect.getX());

int centerY = (int) Math.floor(rect.height / 2 + rect.getY());

Assertions.assertEquals("-1", moveTo.get("button"));

Assertions.assertEquals("pen", moveTo.get("pointerType"));

Assertions.assertEquals(String.valueOf(centerX), moveTo.get("pageX"));

Assertions.assertEquals(String.valueOf(centerY), moveTo.get("pageY"));

Assertions.assertEquals("0", down.get("button"));

Assertions.assertEquals("pen", down.get("pointerType"));

Assertions.assertEquals(String.valueOf(centerX), down.get("pageX"));

Assertions.assertEquals(String.valueOf(centerY), down.get("pageY"));

Assertions.assertEquals("-1", moveBy.get("button"));

Assertions.assertEquals("pen", moveBy.get("pointerType"));

Assertions.assertEquals(String.valueOf(centerX + 2), moveBy.get("pageX"));

Assertions.assertEquals(String.valueOf(centerY + 2), moveBy.get("pageY"));

Assertions.assertEquals("0", up.get("button"));

Assertions.assertEquals("pen", up.get("pointerType"));

Assertions.assertEquals(String.valueOf(centerX + 2), up.get("pageX"));

Assertions.assertEquals(String.valueOf(centerY + 2), up.get("pageY"));

waitForEnterKey("Pen Actions");

end();

}

private Map<String, String> getPropertyInfo(WebElement element) {

String text = element.getText();

text = text.substring(text.indexOf(' ') + 1);

return Arrays.stream(text.split(", ")).map(s -> s.split(": ")).collect(Collectors.toMap(a -> a[0], a -> a[1]));

}

private void end() {

sc.close();

driver.quit();

}

private void waitForEnterKey(String demoName) {

while (true) {

System.out.println("Press enter after taking screenshots of " + demoName + " demo.");

String entered = sc.nextLine();

if (entered.equals("")) {

break;

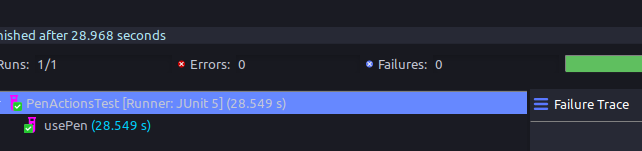
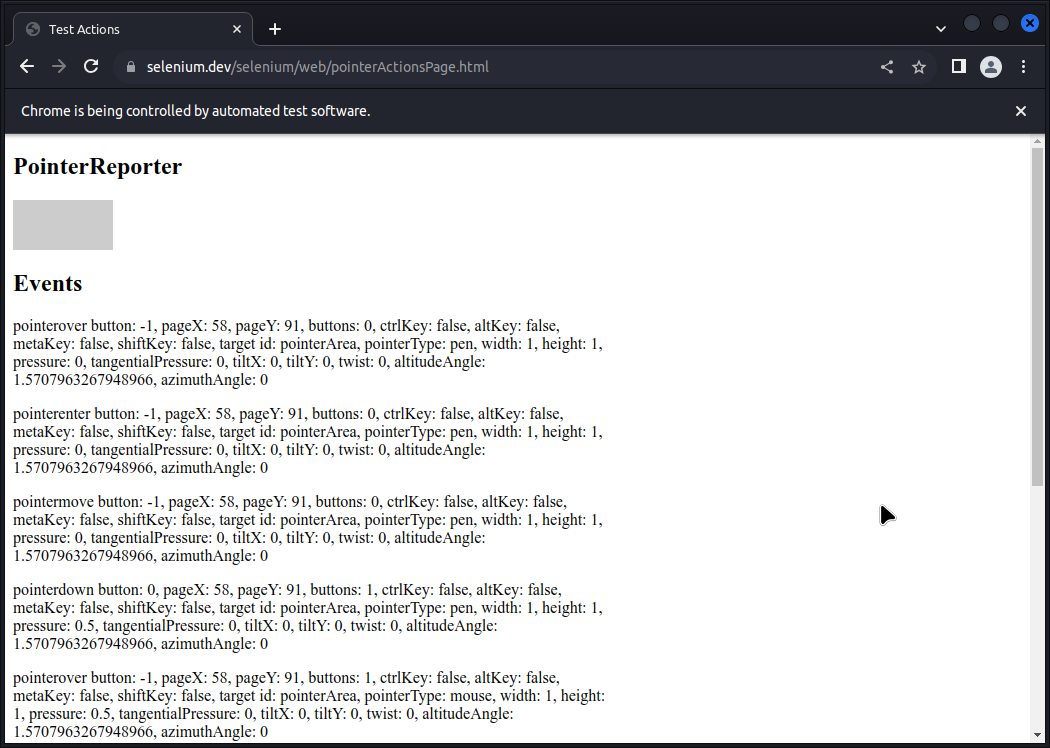
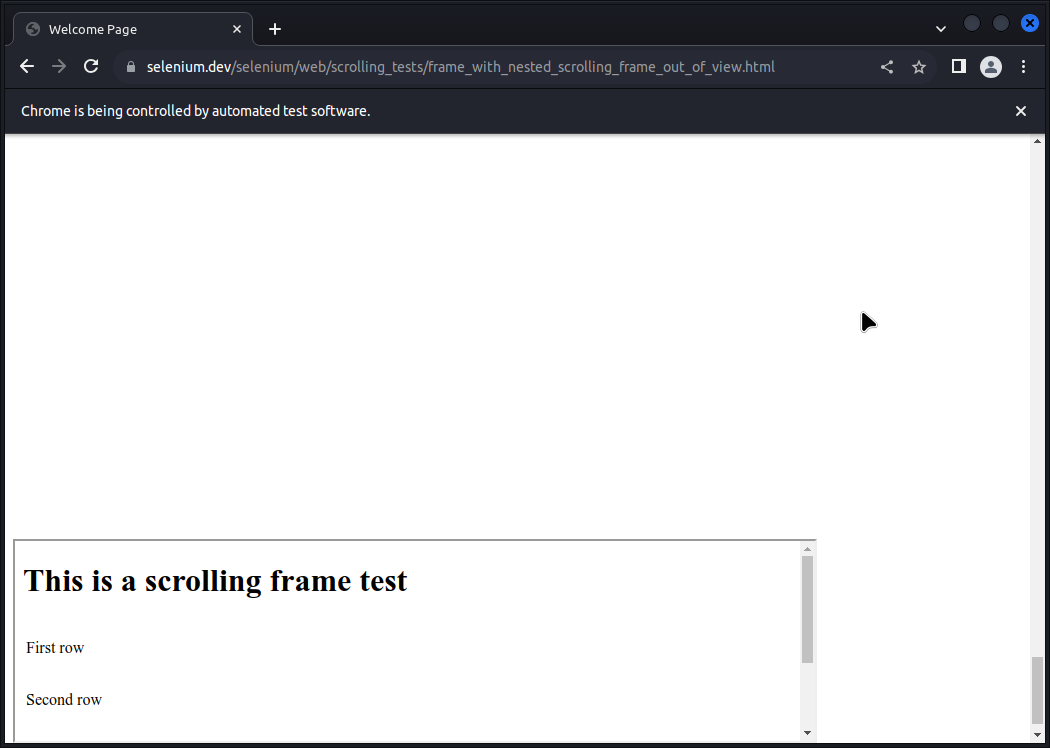
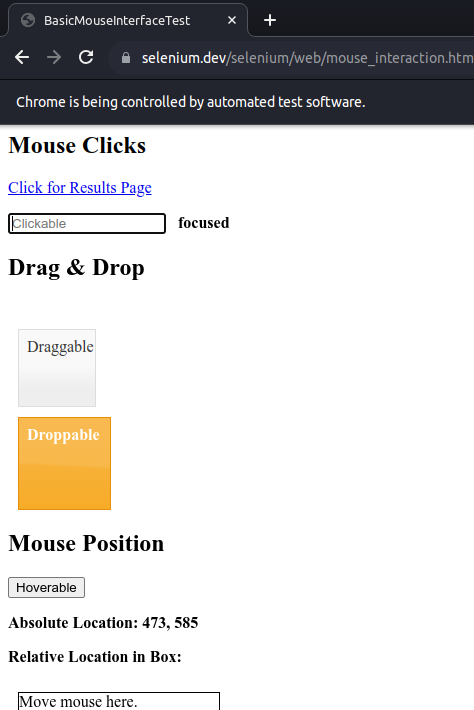
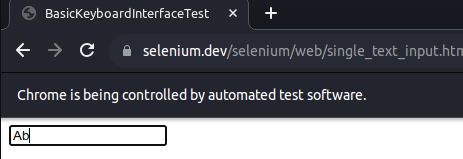
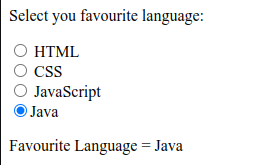
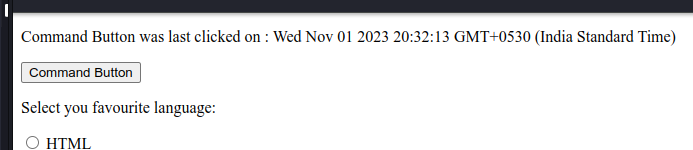
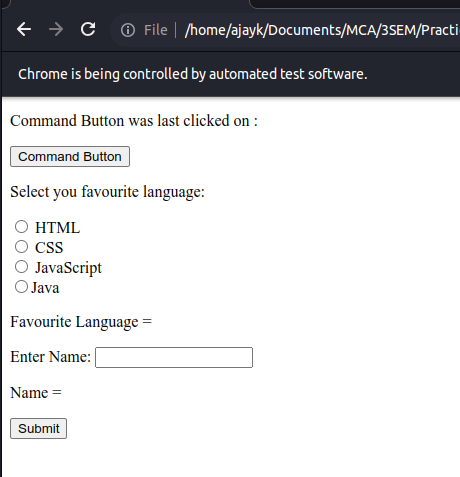
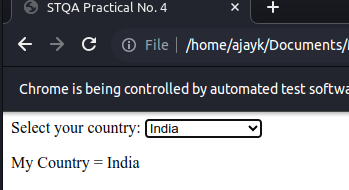
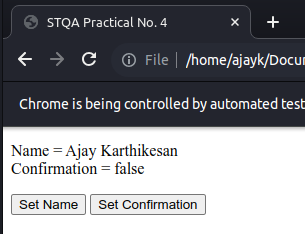
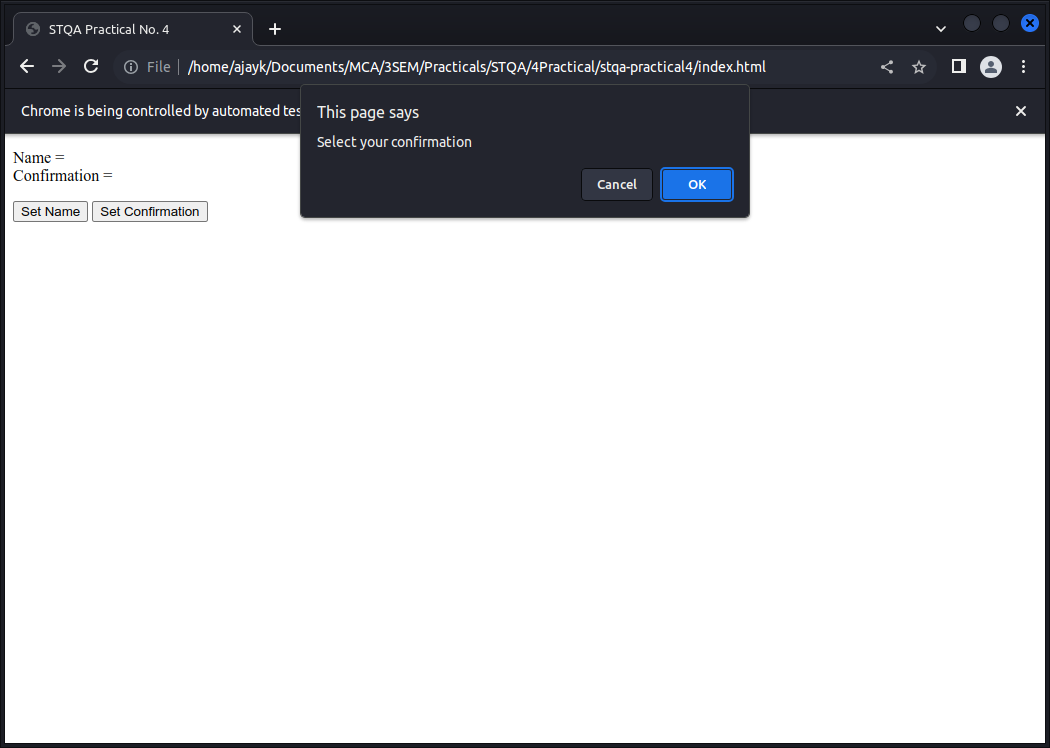
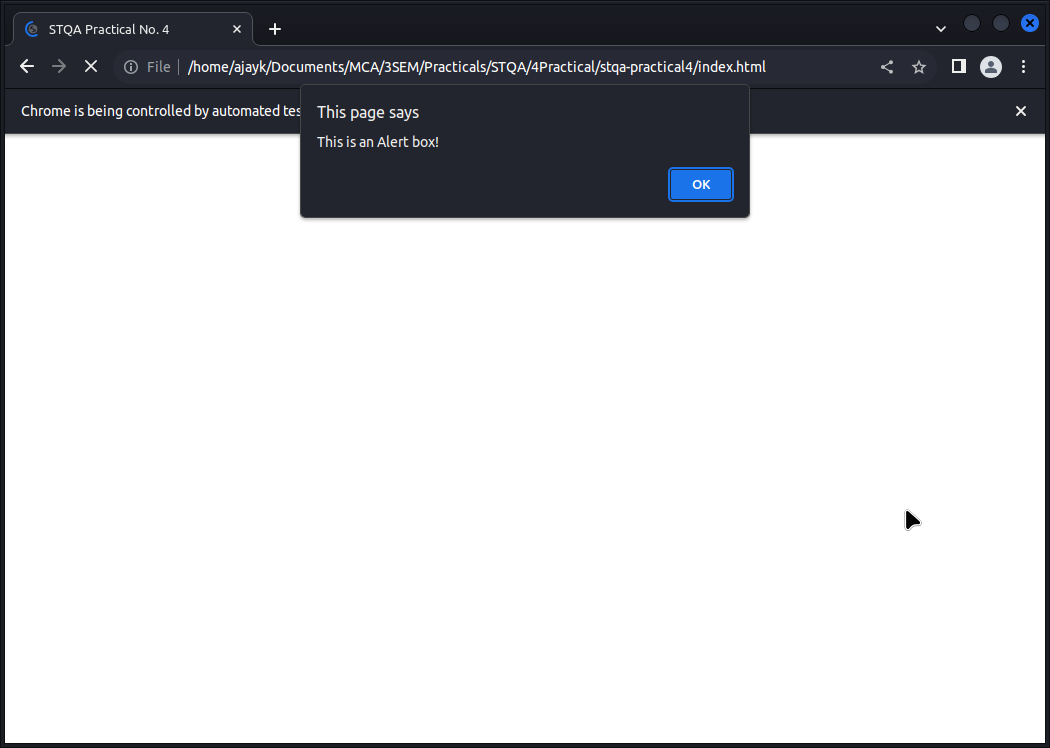
}

}

}

}

**Output:**



**Conclusion:**

I learnt how to…

1.Demonstrate different types of alerts

2. Demonstrate :

Handling Drop Down

List Boxes

3. Demonstrate

Command Button

Radio buttons & text boxes

Waits command in selenium

4. Demonstrate action classes in Selenium