Roll No. **42** Exam Seat No._____

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R. C. Marg,Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr. **CHRISTY PHILIP [ROLL NO: 42]** of **SYMCA-2B** has satisfactorily completed a course of the necessary experiments in **MCAL35 – Software Testing Quality Assurance Lab** under the supervision of **Dr. Meeanakshi Garg** in the Institute of Technology in the academic year **2022- 2023.**

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony, Chembur, Mumbai

Department of M.C.A

MCAL35 – SOFTWARE TESTING QUALITY ASSURANCE LAB INDEX

Sr. No	Contents	Date Of Preparation	Date Of Submission	Marks	Sign
1	Take a review and write test cases for any known application.	22/08/2022	29/08/2022	10	
2	Implement Web Drivers on Chrome & Firefox Browsers.	29/08/2022	05/09/2022	10	
3	Demonstrate handling multiple frames in selenium.	05/09/2022	12/09/2022	10	
4	Implement Browser command and navigation Commands.	12/09/2022	19/09/2022	10	
5	Implement the find element command.	19/09/2022	26/10/2022	10	
6	Demonstrate the Locator. (id, css selector, path)	19/09/2022	26/09/2022	10	
7	Demonstrate synchronization in selenium.	26/09/2022	03/10/2022	10	
8	Demonstrate different types of alerts.	26/09/2022	03/10/2022	10	
9	Demonstrate: Handling Drop Down & List Boxes.	03/10/2022	10/10/2022	10	
10	Demonstrate Command Button Radio buttons & text boxes.	03/10/2022	10/10/2022	10	
11	Demonstrate action classes in Selenium.	10/10/2022	17/10/2022	10	
12	Installation of TestNg, running TestNg and TestNg annotations.	17/10/2022	31/10/2022	10	
13	Demonstrate data driven Framework.	31/10/2022	14/11/2022	10	
A1	Assignment I: Software Basics - I	19/09/2022	26/09/2022	10	
A2	Assignment II: Software Basics – II	19/09/2022	26/09/2022	10	
A3	Assignment III: Software Basics - III	19/09/2022	26/09/2022	10	

PRACTICAL 1

AIM: - Take a review and write test cases for any known application.

THEORY: -

Project Description

MYNTRA SHOPPING WEBSITE

Myntra is famous for its clothing. The Myntra through its online shopping facility serves customers with a large number of brands and selection for both the Gender and for all the ages. Myntra is now owned by Flipkart. Myntra provides so many facilities which makes it user friendly. The website provides many offers to customers. Website is providing the major functionality on which the test cases will be prepared.

1) REGISTRATION: -

The Myntra provides the registration page for each of its new user's The registration is done using phone number and many other details to ensure security. All these details have to be checked so this test case will be designed.

2) LOGIN: -

The registered user is given the facility of Login to access the features of the website. The Login is done using the Phone number with further OTP and Password to ensure security. We will be designing test cases for each item.

3) ORDER: -

The Order facility is provided to the user for ordering of the products. The product order should be seen in the bag. So for this we have to design the test cases.

4) WISHLIST: -

The wishlist feature is provided by Myntra to store the user's favorite items in order to easily access it when needed. So for the user's convenience, Myntra provides a Wishlist section. So we have to design test cases for the wishlist section.

5) PAYMENT: -

The payment section the Myntra uses is of the third party. The payment is a very critical section for the stakeholders. So we have to design the payment test cases with a critical care and covering all the test cases.

6) PROFILE UPDATION: -

The myntra provides the profile updation feature so the user can change if necessary the required details when needed. This can also be misused by the hacker to prevent Myntra from providing step verification. So the test case has to be created for this functionality in a serious manner.

Test Case 1: Verify registration with Details

Description: Test the Myntra Registration Page

Pre-Conditions: User has not yet registered

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to registration page	Data=New User	User should be on the page of Registration	User is on the Registration	Pass
2	Create Password	Password with minimum 8 characters, at least 1 uppercase, at least 1 lowercase and 1 special symbol	Password is correct	Password is correct	Pass
		Characters are less than 8	Please enter at least 8 characters	Please enter at least 8 characters	Pass
		Uppercase is missing	Please enter at least 1 Uppercase character	Please enter at least 1 Uppercase character	Pass
		Special symbol is missing	Please enter at least 1 Special character	Please enter at least 1 Special character	Pass
		Numeric value is missing	Please enter at least 1 Numeric value	Please enter at least 1 Special character	Pass

3	Full Name	Proper Name Entered	Full Name is Accepted	Full Name is Accepted	Pass
		Without entering value	Please Enter Name	Please Enter Name	Pass
4	Gender	Gender was selected	Accepted	Accepted	Pass
		Gender was not selected	Please select Gender	Please select Gender	Pass

Post Conditions:

User is able to register him successfully

Test Case 2: Verify login with Details

Description: Test the Myntra Login Page

Pre-Conditions: User has not yet Logged In

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to Login Page	Date=Login Id	User should be on the page of Login	User is on the Login	Pass
2	Provide Mobile Number	Valid phone number is been provided	User is directed to OTP Page	User is been directed to OTP page	Pass
		Invalid phone number is been provided	Please enter a valid mobile number(10 digits)	Please enter a valid mobile number(10 digits)	Pass
3	OTP Validation	Correct OTP	Logged in successfully	Logged in successfully message is shown	Pass
		Incorrect OTP	Incorrect OTP message shown and login denied	Incorrect OTP message is shown	Pass

Post Conditions:

User is able to Login successfully

Test Case 3: Verify with Orders

Description: Test the Myntra Orders Page

Pre-Conditions: User is Ordering

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	User is clicking on Orders in profile	Click on Profile and then click on Orders	User goes to orders page	User is on orders page	Pass
2	Orders something	User clicks on any order item	The order should be accepted and to be shown in orders section	The order is accepted and to be shown in orders section	Pass
		User checks the orders without ordering anything	The order section is to be shown empty	The order section is empty	Pass
3	Selecting size	User selects the size	He should be able to click the add to bag without any message	User was able to click the add to bag	Pass
		User clicks on add to bag without selecting the size	User is shown message Please select a size	User was shown the message please select a size	Pass

Post Conditions:

User is able to Order his item successfully

Test Case 4: Verify with Wishlist

Description: Test the Myntra Wishlist Page

Pre-Conditions: User is on any of the Products

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	User is clicking on the wishlist	User is clicking the wishlist icon	User should be shown wishlist	User was directed to the wishlist on clicking icon	Pass
2	User is on any product	User clicks on the wishlist	The item should be added to the wishlist	User's item is added to the wishlist	Pass
		User doesn't click the wishlist	The item shouldn't be added to the wishlist	The item is not been shown in wishlist	Pass
3	User is clicking on the wishlisted item	User clicks the item he has wishlisted early	There should be no adding of same item again	The item isn't added to wishlist	Pass
4	User is on the wishlist list	User clicks cancel icon on the wishlist item	The item should be removed	The item is removed	Pass

Post Conditions:

User is able to add the items successfully to his other wishlist.

Test Case 5:Payment

Description: Test the Payment

Pre-Conditions: User is paying the cost of the Product

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	User places the order	Address properly entered	Accept the address	Accepted	Pass
2	User places the order	Address without landmark	Message is shown proper landmark and address to be entered	Message of proper entering of address was shown	Pass
3	Payment with UPI	User chooses UPI and enters valid UPI	UPI is accepted	UPI is verified	Pass
		User chooses UPI and enters invalid UPI	UPI is not accepted	UPI is not verified	Pass
4	Payment with Net Banking	Enter valid User ID	User ID is accepted	User ID is verified	Pass
		Enter invalid User ID	User ID is not accepted	User ID is not verified	Pass
5	Card Payment	Correct Card Number is been entered	Card number is verified	Card is accepted	Pass

		Incorrect Card Number is been entered	Card number is not verified	Card is not accepted	Pass
6	Order should be placed after payment	Payment is done successfully	Order is placed	Order is placed	Pass

Post Conditions:

User is able to make payment in successful and in a easy manner

Test Case 6:Profile Updation

Description: Test the Profile Updation

Pre-Conditions: User is updating the profile

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Change phone number is clicked	The user clicks on change	OTP should be asked as part of 2 step verification	OTP is asked	Pass
2	Changing the name	The user clears the Name	OTP should be asked as part of 2 step verification	OTP is asked	Pass
3	User clicks on change password	The password entering box is needed	Old and new password entering box should appear as prompt	Old and new password entering box is asked as prompt	Pass

Post Conditions:

User was able to update the profile successfully

CONCLUSION: -

The usability evaluation of website design(Myntra) has been implemented successfully

PRACTICAL 2

AIM: - Implement Web Drivers on Chrome & Firefox Browsers.

THEORY: -

WebDriver is a tool for automating testing web applications. It is popularly known as Selenium 2.0. WebDriver uses a different underlying framework, while Selenium RC uses JavaScript Selenium-Core embedded within the browser which has some limitations. WebDriver interacts directly with the browser without any intermediary, unlike Selenium RC that depends on a server. It is used in the following context −

Multi-browser testing including improved functionality for browsers which is not well-supported by Selenium RC (Selenium 1.0).

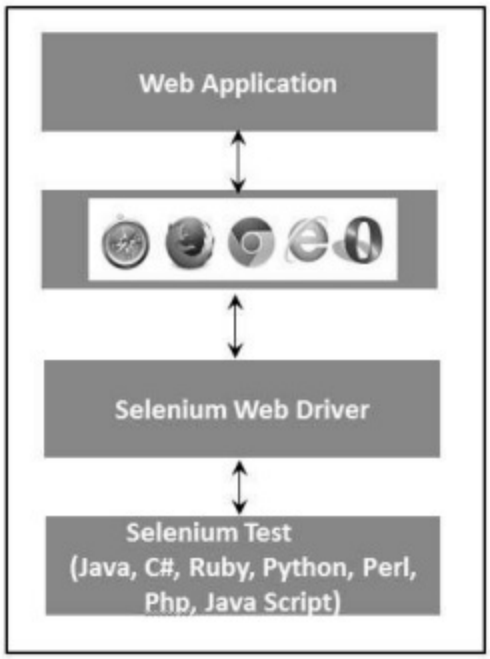
Handling multiple frames, multiple browser windows, popups, and alerts.

Complex page navigation.

Advanced user navigation such as drag-and-drop.

AJAX-based UI elements.

Architecture
WebDriver is best explained with a simple architecture diagram as shown below.



Steps: -

1) Download Selenium Driver from https://www.selenium.dev/downloads/

2) Download Required Browser Drivers according to Browser versions.

3) Add jar files of Selenium Driver in Netbeans Project(Inside Lib folder as well as outside)

4) Path of both driver should be noted

5) Open Netbeans

6) Add appropriate driver path in code

CODE: -

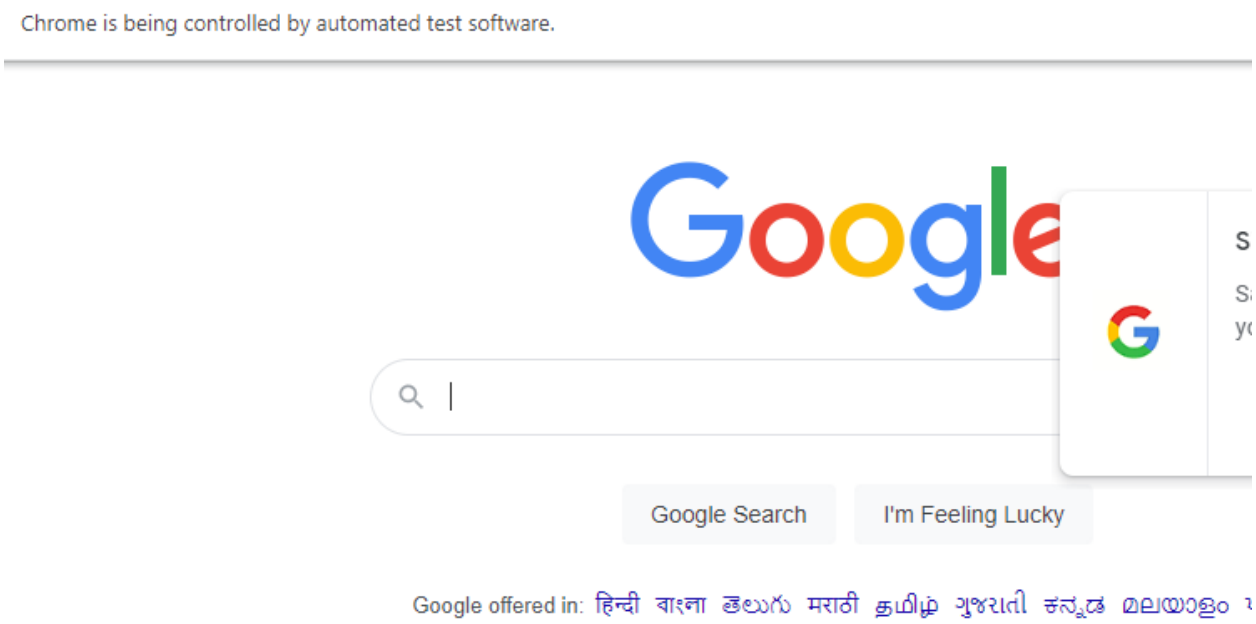
```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package stqa2;
import java.util.Scanner;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
```

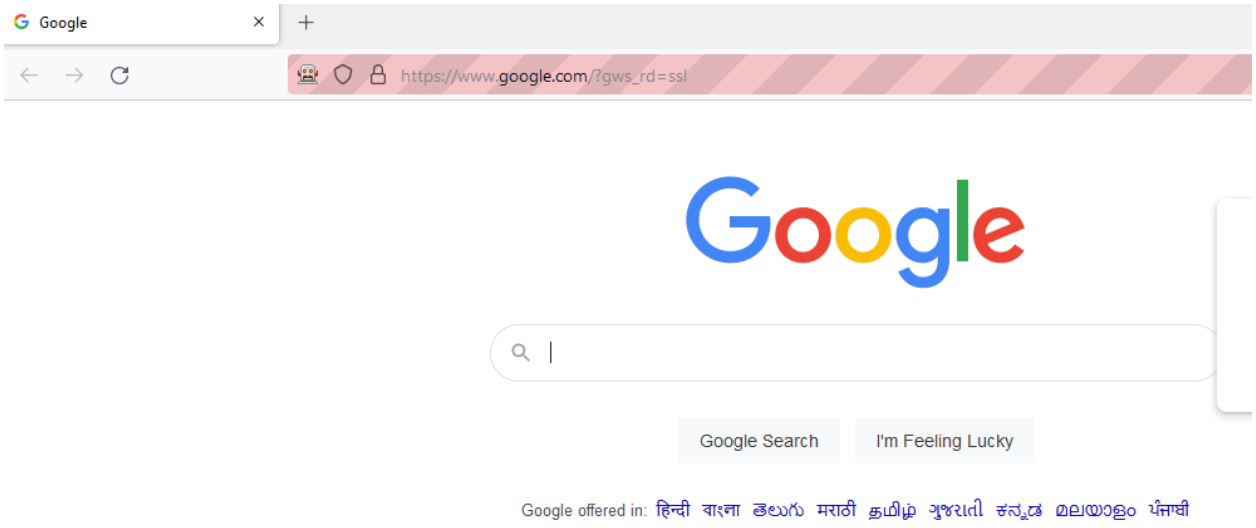
```
import org.openqa.selenium.firefox.FirefoxDriver;
/**
 *
 * @author user
 */
public class Stqa2 {

   static WebDriver wd;
        public static void main(String[] args) {

System.setProperty("webdriver.chrome.driver","C:\\Users\\user\\Downloads\\chromedriver_win32\\chromedriver.exe");
                Scanner sc = new Scanner(System.in);
                System.out.println("1.ChromeBrowser \n2. Firefox Browser");
                System.out.println("Choice");
                int ch = sc.nextInt();
                sc.close();
                switch (ch)
                {
                case 1:

System.setProperty("webdriver.chrome.driver","C:\\Users\\user\\Downloads\\chromedriver_win32\\chromedriver.exe");
                        wd = new ChromeDriver();
                        break;
                case 2:

System.setProperty("webdriver.gecko.driver","C:\\Users\\user\\Downloads\\geckodriver-v0.31.0-win64\\geckodriver.exe");
                        wd = new FirefoxDriver();


                default:
                        System.out.println("Invalid Choice");
                        break;
                }
                if(wd!=null)
                {
                        wd.get("http:\\google.com");
                }
```

```
        }

}
```

OUTPUT: -

```
1.ChromeBrowser
2. Firefox Browser
Choice
```





CONCLUSION: -

From this practical I have learned implementation of Web Drivers on Chrome & Firefox Browsers.

PRACTICAL 3

AIM: - Demonstrate handling multiple frames in selenium

THEORY: -

iFrame in Selenium Webdriver

iFrame in Selenium Webdriver is a web page or an inline frame which is embedded in another web

page or an HTML document embedded inside another HTML document. The iframe is often used to

add content from other sources like an advertisement into a web page. The iframe is defined with the <iframe> tag.

We can identify the frames in Selenium using methods given below:

- Right click on the element, if you find the option like 'This Frame' then it is an iframe. (Please refer the below steps)
- Right click on the page and click 'View Page Source' and Search with the 'iframe', if you can find any tag name with the 'iframe' then it means to say the page consists of an iframe.

In the above diagram, you can see that 'This Frame' option is available upon right clicking, so we are now sure that it is an iframe.

We can even identify the total number of iframes by using the below code:

```
Int size = driver.findElements(By.tagName("iframe")).size();
```

Steps: -

1) Search Ctrl+Shift+i by going on a website

2) Search by clicking Ctrl+f iframe

3) Choose the appropriate id and use it in the code.

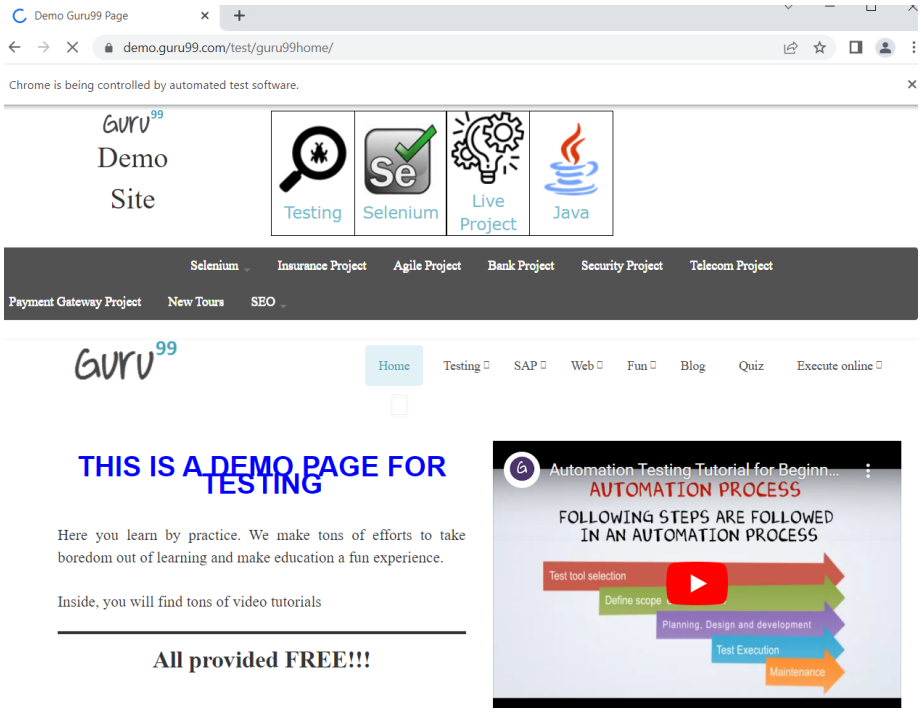
Link: -
http://demo.guru99.com/test/guru99home/

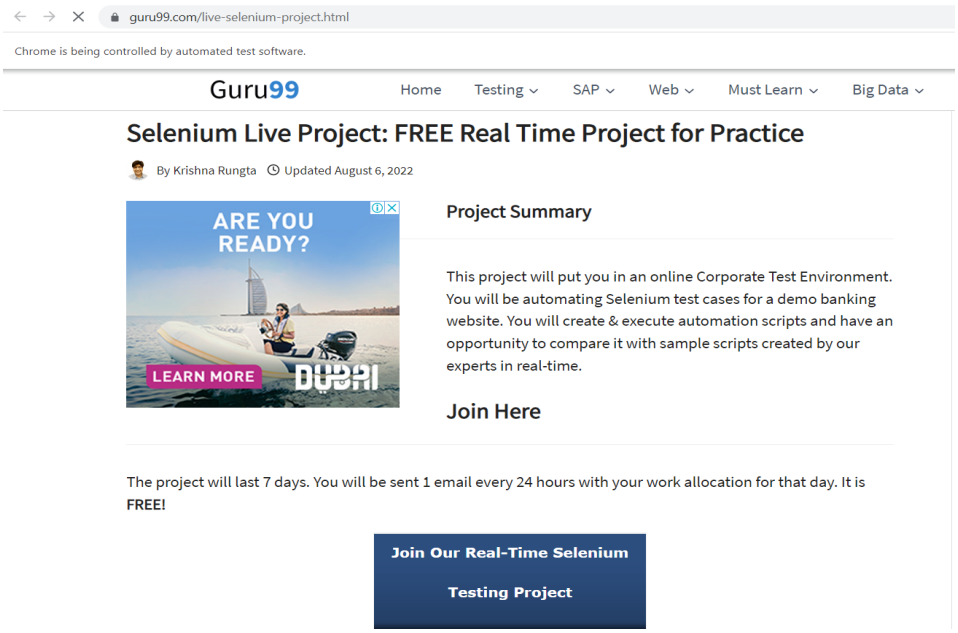
CODE: -

```
package stqa3;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

/**
 *
 * @author Philip
 */
public class Stqa3 {

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


System.setProperty("webdriver.chrome.driver","D:\\WEB_BROWSER_DRIVER\\chromedriver
_win32\\chromedriver.exe");

                driver = new ChromeDriver();
                driver.get("http://demo.guru99.com/test/guru99home/");

                // navigates to the page consisting an iframe
                driver.manage().window().maximize();
                Thread.sleep(2000);

                driver.switchTo().frame("a077aa5e"); //switching the frame by ID
                Thread.sleep(2000);

                System.out.println("Switching to iframe");
                driver.findElement(By.xpath("html/body/a/img")).click();
                Thread.sleep(2000);

                //Clicks the iframe
```

```
            System.out.println("Finished");
        }
}
```

OUTPUT: -





```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 106.0.5249.21 (71f4e2c9a6f38f5550a8ea29a984a08304554956-refs/branch-heads/5249@{#228}) on port 59479
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Oct 02, 2022 9:28:15 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 02, 2022 9:28:15 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 02, 2022 9:28:15 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Switching to iframe
Finished
BUILD SUCCESSFUL (total time: 25 seconds)
```

CONCLUSION: -

From this practical I have learned to demonstrate handling multiple frames in selenium.

PRACTICAL 4

AIM: - Implement Browser command and navigation Commands.

THEORY: -

Selenium WebDriver - Browser Commands:

The very basic browser operations of WebDriver include opening a browser; perform few tasks and then closing the browser.

Given are some of the most commonly used Browser commands for Selenium WebDriver.

1. Get Command:

Method: get(String arg0) : void

In WebDriver, this method loads a new web page in the existing browser window. It accepts String as parameter and returns void.

Implementation:

```
driver.getTitle();
OR
String Title = driver.getTitle();
```

2. Get Title Command:

Method: getTitle(): String

In WebDriver, this method fetches the title of the current web page. It accepts no parameter and returns a String.

Implementation:

```
driver.getTitle();
OR
String Title = driver.getTitle();
```

3. Get Current URL Command:

Method: getCurrentUrl(): String

In WebDriver, this method fetches the string representing the Current URL of the current web page. It accepts nothing as parameter and returns a String value.

Implementation:

```
driver.getCurrentUrl();
OR
String CurrentUrl = driver.getCurrentUrl();
```

4. Close Command:

Method: close(): void

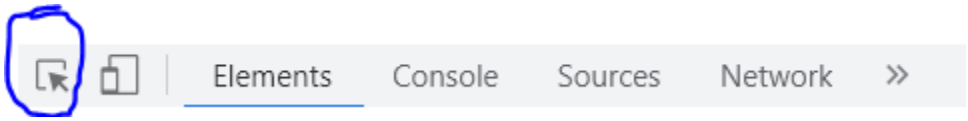
This method terminates the current browser window operating by WebDriver at the current time. If the current window is the only window operating by WebDriver, it terminates the browser as well. This method accepts nothing as parameter and returns void.

Implementation:

```
driver.close();
```

Steps: -

1) Goto site of your choice and do Ctrl+Shift+i
2) Click on the circled icon in the image below then on any clickable like Log In, Sign Up or Contact Us.
3) The left side will show lines based on clicked Option.
4) Goto the line
5) Right click on it hover mouse on copy
6) Copy full Xpath

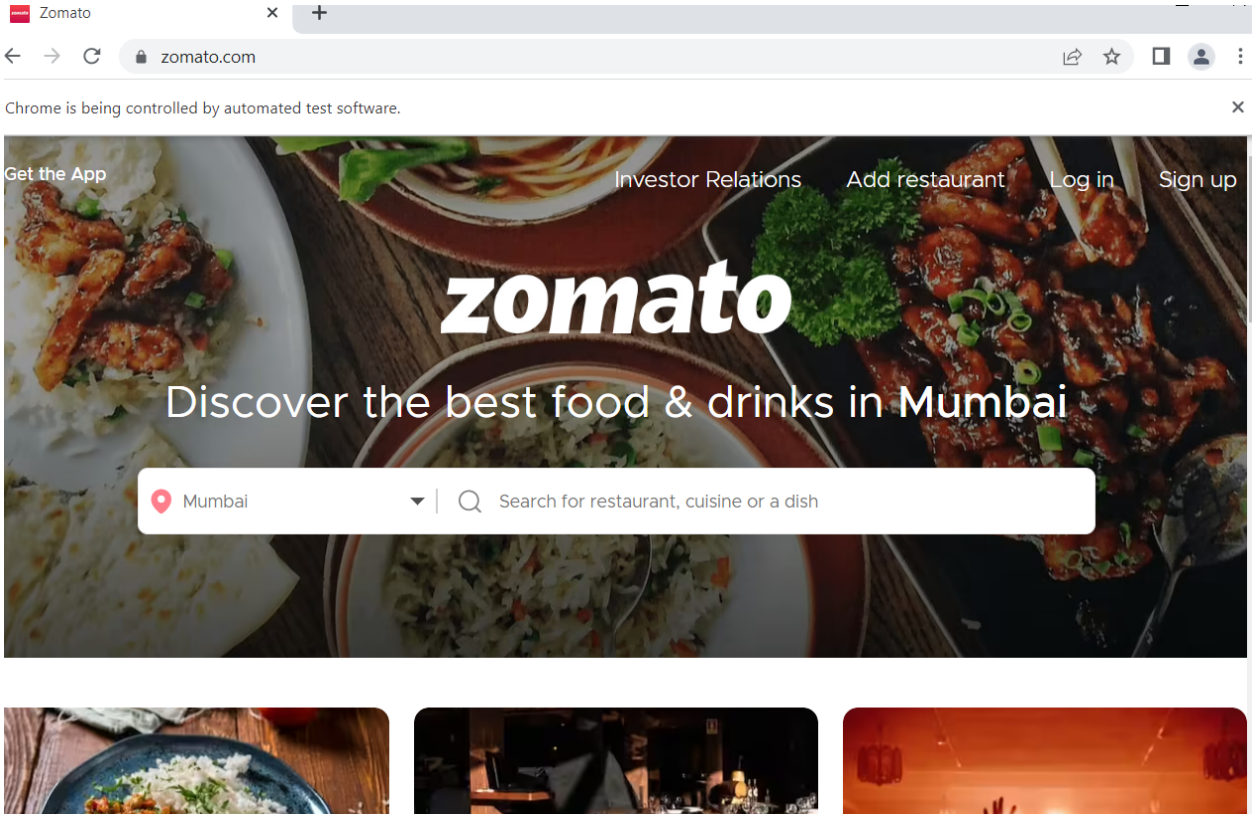


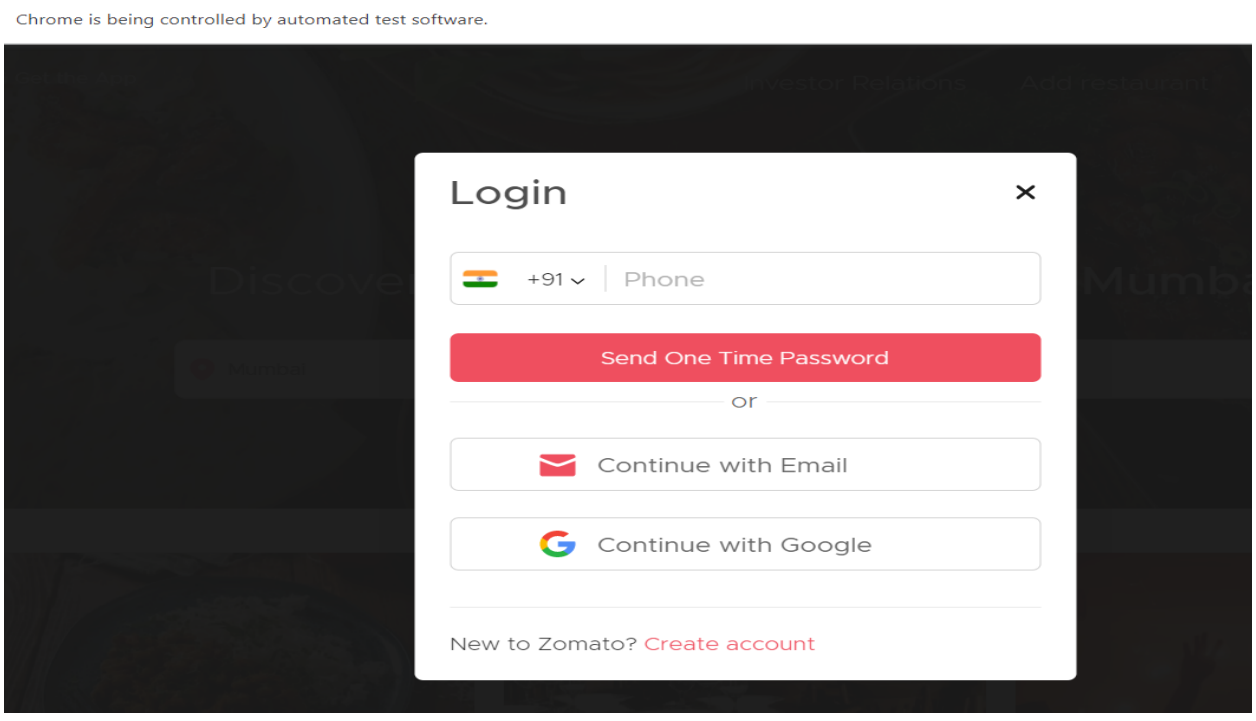
CODE: -

```
package stqa4;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
/**
 *
 * @author Philip
 */
public class Stqa4 {
```

```
static WebDriver driver;
	public static void main(String[] args) throws InterruptedException {
	// TODO code application logic here

System.setProperty("webdriver.chrome.driver","D:\\WEB_BROWSER_DRIVER\\chromedriver
_win32\\chromedriver.exe");
	driver = new ChromeDriver();
	String appUrl = "https://www.zomato.com/";
	driver.get(appUrl);
	Thread.sleep(3000);
	// Click on Contact Us

driver.findElement(By.xpath("/html/body/div[1]/div/div[2]/header/nav/ul[2]/li[4]/a")).click();
	Thread.sleep(3000);
	// Go back to Home Page
	driver.navigate().back();
	Thread.sleep(3000);
	// Go forward to Contact Us
	driver.navigate().forward();
	Thread.sleep(3000);
	// Go back to Home page
	driver.navigate().to(appUrl);
	Thread.sleep(3000);
	// Refresh browser
	driver.navigate().refresh();
	Thread.sleep(3000);
	// Close browser
	driver.close();
	}

}
```

OUTPUT: -


Zomato
zomato.com
Chrome is being controlled by automated test software.
Get the App
Investor Relations
Add restaurant
Log in
Sign up
zomato
Discover the best food & drinks in Mumbai
Mumbai
Search for restaurant, cuisine or a dish


Chrome is being controlled by automated test software.
Login
+91
Phone
Send One Time Password
or
Continue with Email
Continue with Google
New to Zomato? Create account

```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 106.0.5249.21 (71f4e2c9a6f38f5550a8ea29a984a08304554956-refs/branch-heads/5249@{#228}) on port 60518
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Oct 02, 2022 9:46:51 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 02, 2022 9:46:52 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 02, 2022 9:46:52 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
BUILD SUCCESSFUL (total time: 31 seconds)
```

CONCLUSION: -

From this practical I have learned the implementation of Browser command and navigation Commands.

PRACTICAL 5

AIM: - Implement the find element command

THEORY: -

1) FindElement:

Selenium Find Element command takes in the By object as the parameter and returns an object of type list WebElement in Selenium. By object in turn can be used with various locator strategies such as find element by ID Selenium, Name, Class Name, XPATH etc.
Below is the syntax of FindElement command in Selenium web driver:

```
WebElement elementName =
driver.findElement(By.LocatorStrategy("LocatorValue"));
```

Locator Strategy can be any of the following values.

- ID
- Selenium find element by Name
- Class Name
- Tag Name
- Link Text
- Partial Link Text
- XPATH

Locator Value is the unique value using which a web element can be identified. It is the responsibility of developers and testers to make sure that web elements are uniquely identifiable using certain properties such as ID or name.

2) FindElements:

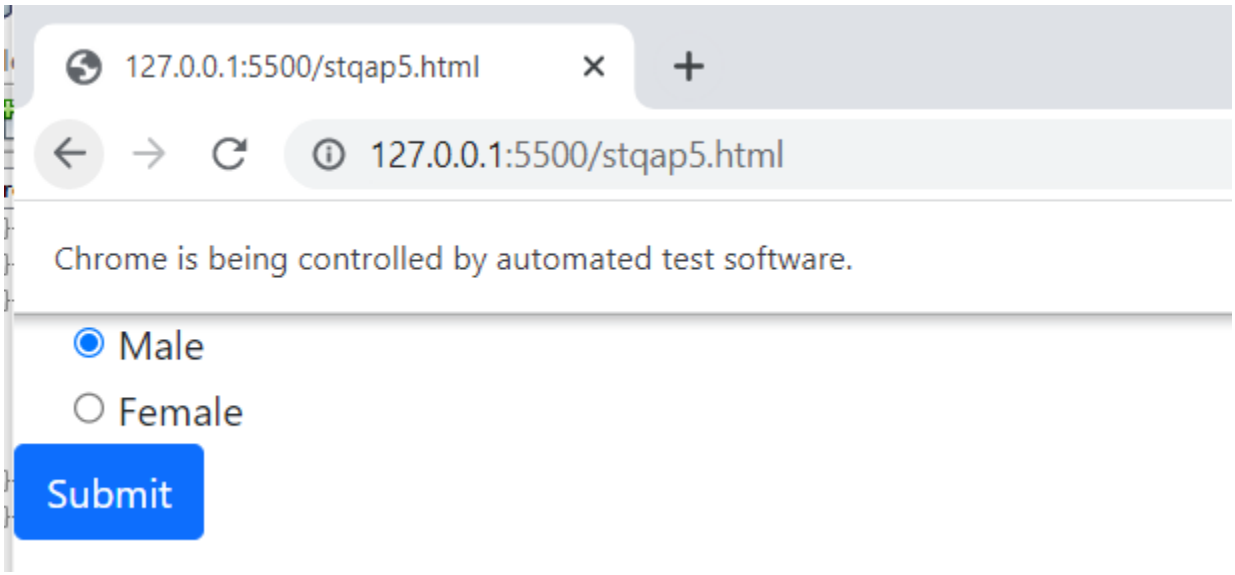
FindElements in Selenium command takes in By object as the parameter and returns a list of web elements. It returns an empty list if there are no elements found using the given locator strategy and locator value.
Below is the syntax of find elements command.

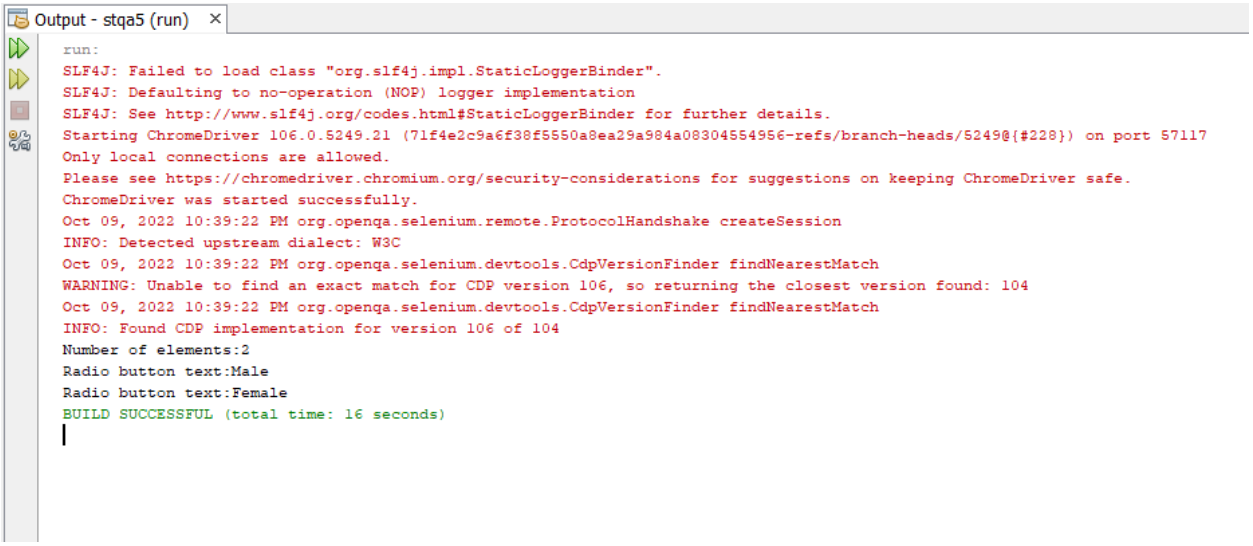
```
List<WebElement> elementName =
driver.findElements(By.LocatorStrategy("LocatorValue"));
```

CODE: -

```
package stqa5;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
/**
 *
 * @author Philip
 */
public class Stqa5 {

   static WebDriver wd;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");
wd = new ChromeDriver();
wd.get("http://127.0.0.1:5500/stqap5.html");
Thread.sleep(2000);
wd.findElement(By.id("flexRadioDefault1")).click();
Thread.sleep(2000);
wd.findElement(By.id("buttoncheck")).click();
Thread.sleep(2000);
List<WebElement> elements = wd.findElements(By.name("gender"));
System.out.println("Number of elements:" + elements.size());
Thread.sleep(2000);
for (int i = 0; i < elements.size(); i++) {
System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));
Thread.sleep(2000);
}
}
}
```

OUTPUT: -





Note: -

For executing the practical an html page should be run on an editor I have used VS Code and Live Server extension. After running the code copy paste URL in Java code. Also all the components should have the id same.

CONCLUSION: -

From this practical I have learned to implement the find element command.

PRACTICAL 6

AIM: - Demonstrate the Locator(id,css selector, path)

THEORY: -

What are Locators?

Locator is a command that tells Selenium IDE which GUI elements (say Text Box, Buttons, Check Boxes etc.) it needs to operate on. Identification of correct GUI elements is a prerequisite to creating an automation script. But accurate identification of GUI elements is more difficult than it sounds. Sometimes, you end up working with incorrect GUI elements or no elements at all! Hence, Selenium provides a number of Locators to precisely locate a GUI element

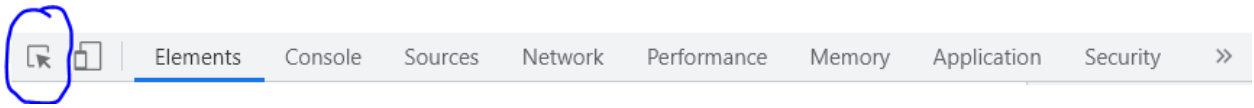
Locator Strategy can be any of the following values.

- ID
- Selenium find element by Name
- Class Name
- Tag Name
- Link Text
- Partial Link Text
- XPATH

Steps: -

1) Goto site https://www.saucedemo.com/

2) By doing Ctrl+Shift+I choose below circled icon



3) Choose the item on the website and note it's Id

id()-contains id

sendKeys-contain parameters

3) We will also need the xpath by right clicking on component→right click→hoer on copy→copy full xpath, it should be the other clickable after the buton

Note: -

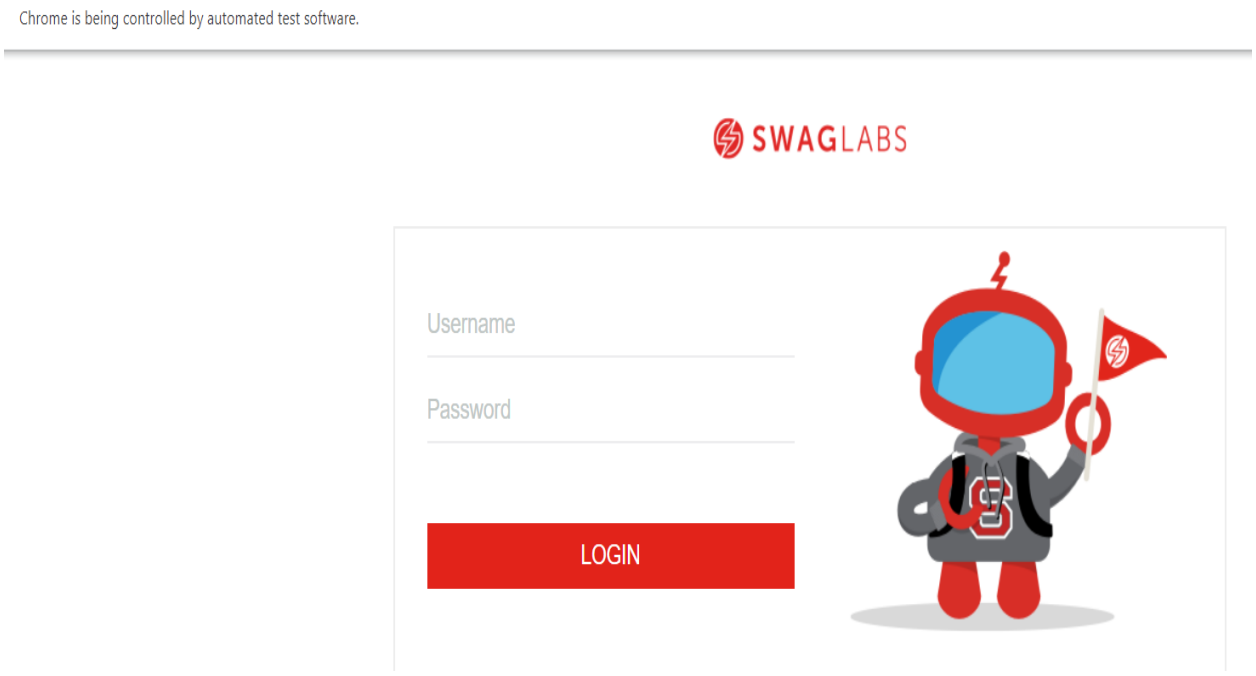
The process may differ based on the websites.

CODE:

```
package stqa6;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
/**
 *
 * @author Philip
 */
public class Stqa6 {

    /**
     * @param args the command line arguments
     */
static WebDriver wd;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");

wd = new ChromeDriver();
wd.get("https://www.saucedemo.com/");
wd.manage().window().maximize();
Thread.sleep(5000);
wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
wd.findElement(By.name("user-name")).sendKeys("standard_user");
Thread.sleep(2000);
wd.findElement(By.id("password")).sendKeys("secret_sauce");
Thread.sleep(2000);
wd.findElement(By.id("login-button")).click();
Thread.sleep(2000);

wd.findElement(By.xpath("/html/body/div/div/footer/ul/li[1]/a")).click();

Thread.sleep(2000);
```

```
wd.navigate().to("https://saucelabs.com/");

Thread.sleep(2000);

}
}
```

OUTPUT: -



Swag Labs

saucedemo.com

Chrome is being controlled by automated test software.

SWAGLABS

standard_user

••••••••••••

LOGIN

Swag Labs

saucedemo.com/inventory.html

Chrome is being controlled by automated test software.

SWAGLABS

PRODUCTS

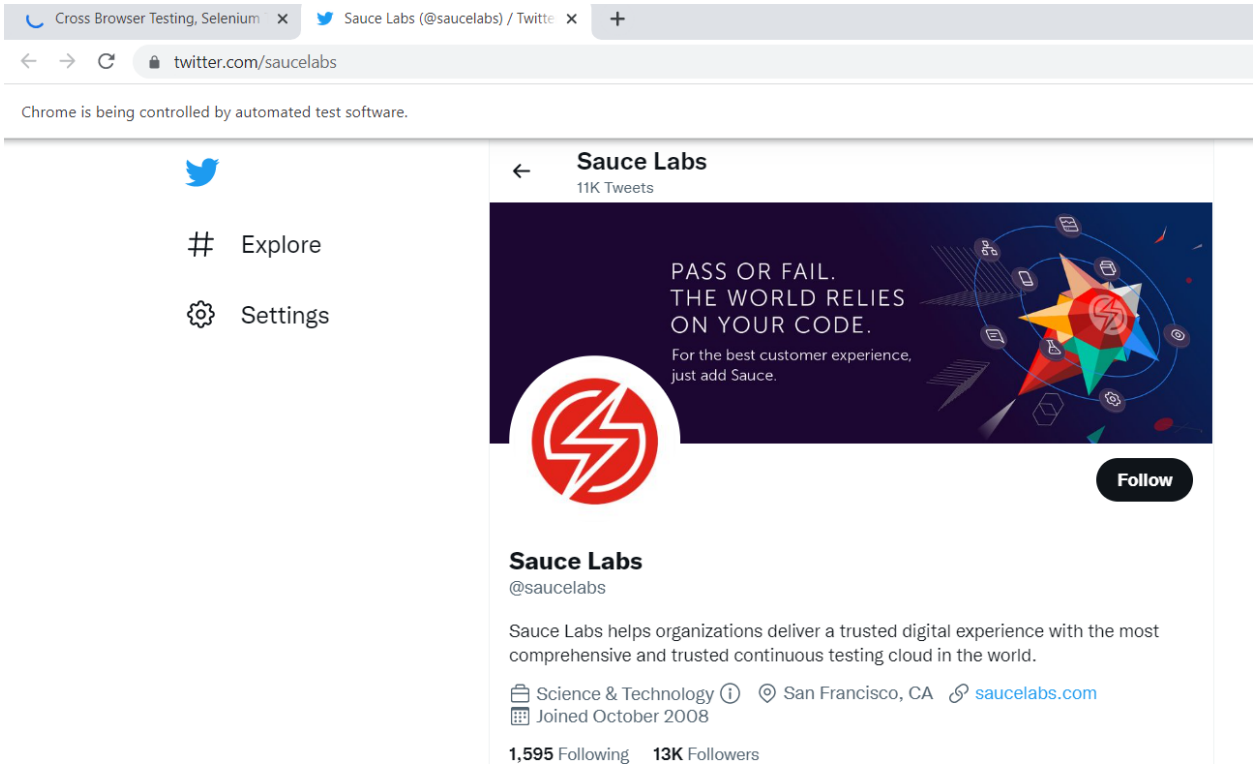
Sauce Labs Backpack

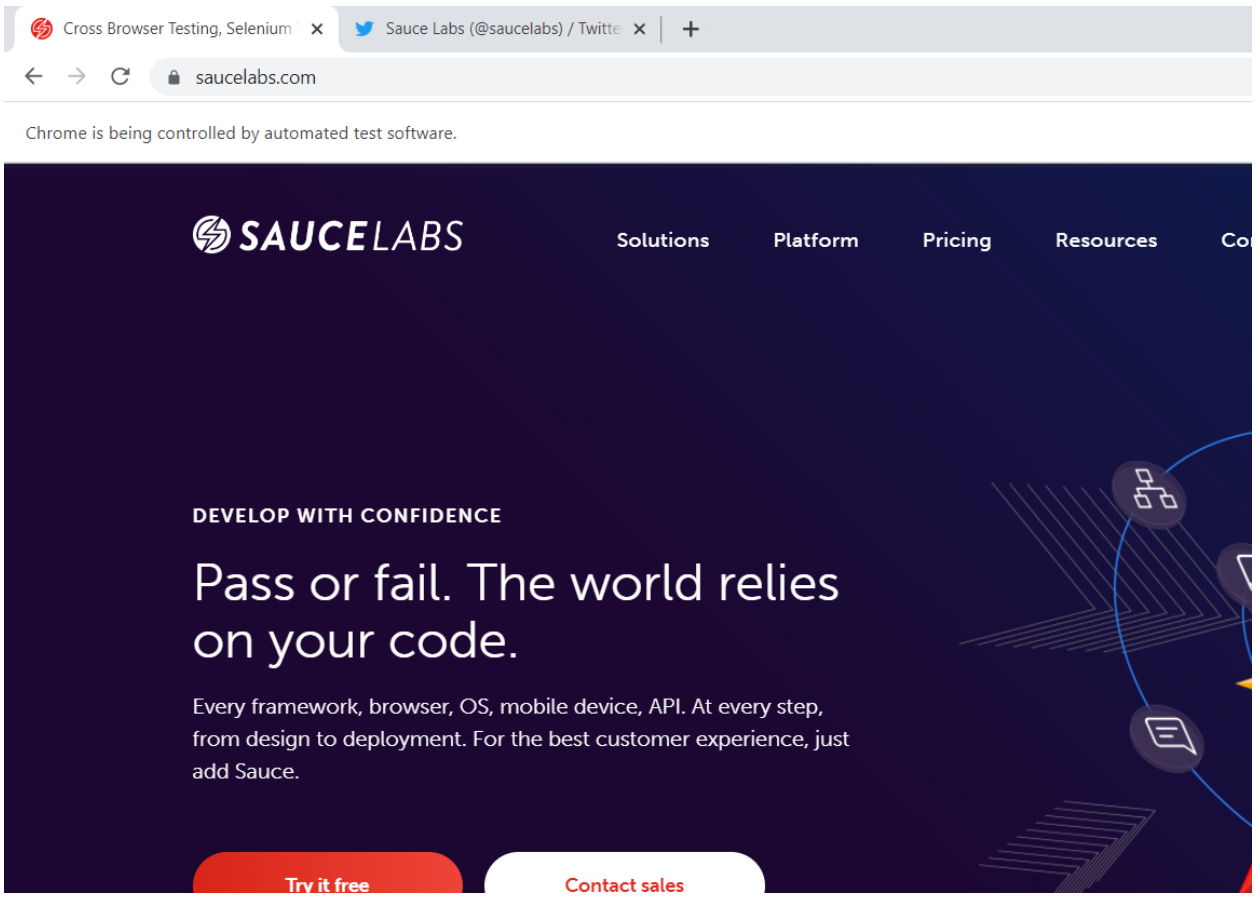
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.

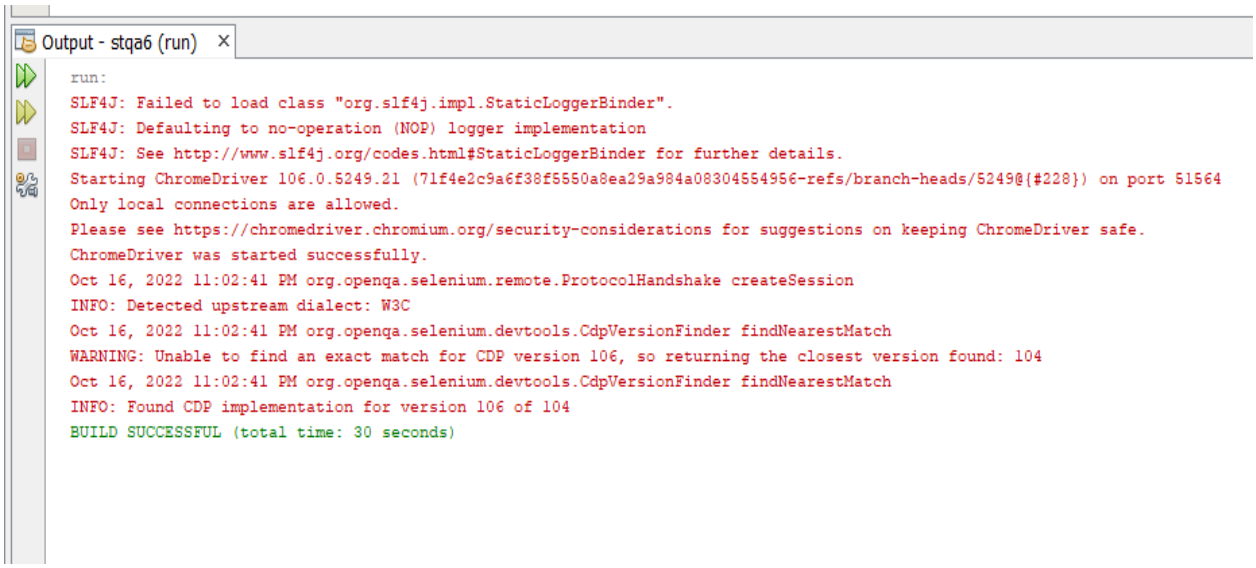
$29.99

ADD TO CART

Sauce Labs Bolt T-Shirt


Cross Browser Testing, Selenium
Sauce Labs (@saucelabs) / Twitte
twitter.com/saucelabs
Chrome is being controlled by automated test software.
Explore
Settings
Sauce Labs
11K Tweets
PASS OR FAIL.
THE WORLD RELIES
ON YOUR CODE.
For the best customer experience,
just add Sauce.
Follow
Sauce Labs
@saucelabs
Sauce Labs helps organizations deliver a trusted digital experience with the most comprehensive and trusted continuous testing cloud in the world.
Science & Technology
San Francisco, CA
saucelabs.com
Joined October 2008
1,595 Following
13K Followers


Cross Browser Testing, Selenium
Sauce Labs (@saucelabs) / Twitte
saucelabs.com
Chrome is being controlled by automated test software.
SAUCELABS
Solutions
Platform
Pricing
Resources
DEVELOP WITH CONFIDENCE
Pass or fail. The world relies on your code.
Every framework, browser, OS, mobile device, API. At every step, from design to deployment. For the best customer experience, just add Sauce.
Try it free
Contact sales




```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 106.0.5249.21 (71f4e2c9a6f38f5550a8ea29a984a08304554956-refs/branch-heads/5249@{#228}) on port 62994
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Oct 17, 2022 6:30:15 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 17, 2022 6:30:15 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 17, 2022 6:30:15 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Test Failed
[1665968419.930][WARNING]: Timed out connecting to Chrome, retrying...
[1665968427.998][WARNING]: Timed out connecting to Chrome, retrying...
BUILD SUCCESSFUL (total time: 23 seconds)
```

CONCLUSION: -

From this practical I have learned to implement demonstration of the Locator(id,css selector, path).

PRACTICAL 7

AIM: - Demonstrate synchronization in selenium

THEORY: -

To synchronize between script execution and application, we need to wait after performing appropriate actions. Let us look at the ways to achieve the same.

Thread.Sleep:

Thread.Sleep is a static wait and it is not a good way to use in scripts as it is sleep without condition.

Thread.Sleep(1000); //Will wait for 1 second.

Explicit Waits:

An 'explicit wait,' waits for a certain condition to occur before proceeding further. It is mainly used when we want to click or act on an object once it is visible.

WebDriver driver = new FirefoxDriver(); driver.get("Enter an URL"S); WebElement DynamicElement =(new WebDriverWait(driver,10)).until(ExpectedConditions.presenceOfElementLocated(By.id("DynamicElement")));

Implicit wait:

Implicit wait is used in cases where the WebDriver cannot locate an object immediately because of its unavailability. The WebDriver will wait for a specified implicit wait time and it will not try to find the element again during the specified time period.

Once the specified time limit is crossed, the webDriver will try to search the element once again for one last time. Upon success, it proceeds with the execution; upon failure, it throws exception.

It is a kind of global wait which means the wait is applicable for the entire driver. Hence, hardcoding this wait for longer time periods will hamper the execution time.

WebDriver driver = new FirefoxDriver(); driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("Enter an URL");

IMPLICIT WAIT

Steps: -

1) Title needs to be noted this is got by doing Ctrl+Shift+I in title part or by hovering the opened tab.

2) This is supplied in eTitle variable.

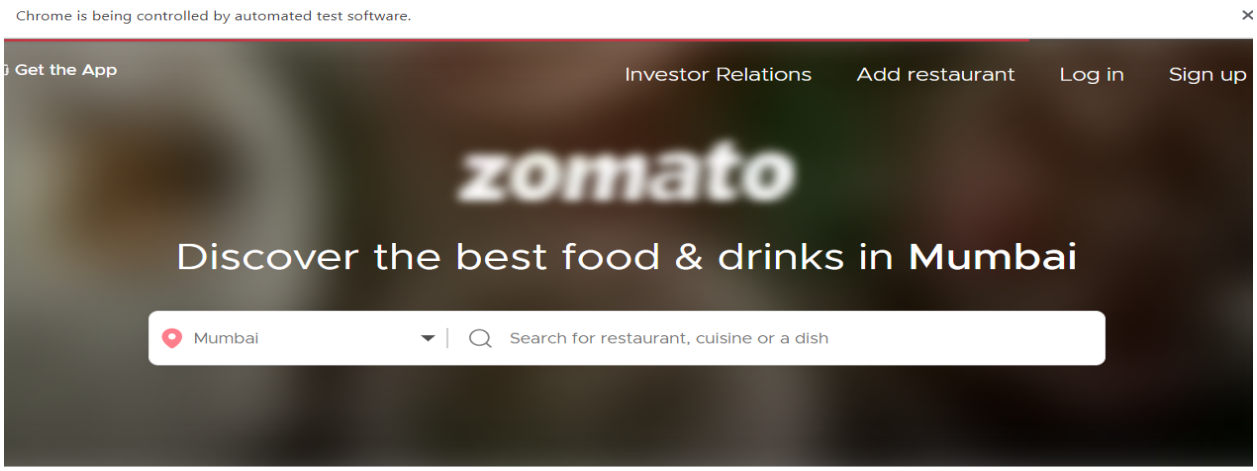
3) If title not properly given Test Failed message shown if properly provided Test Passed message is shown.

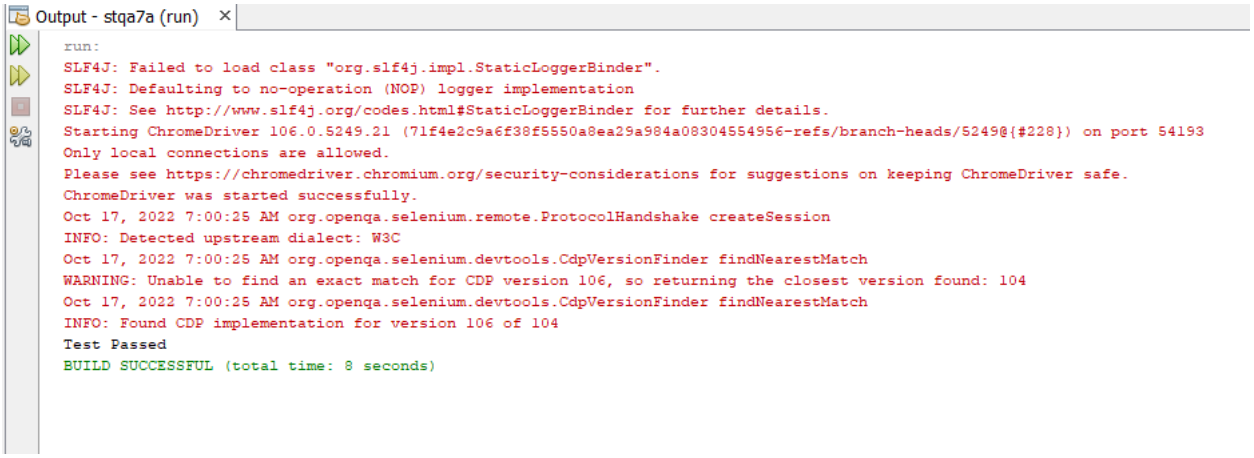
CODE: -

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package stqa7a;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
/**
 *
 * @author Philip
 */
public class Stqa7a {

   static WebDriver driver;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");
driver = new ChromeDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
```

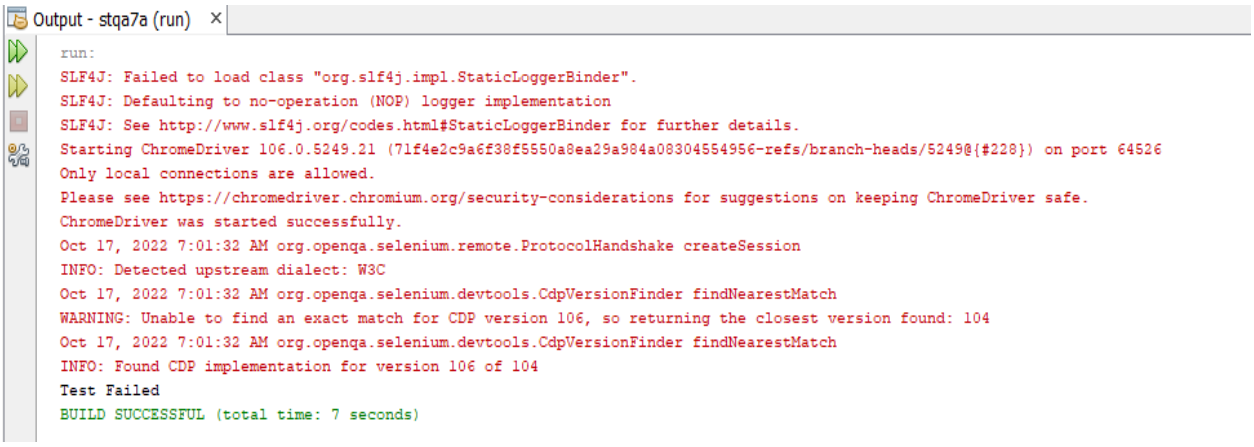
```
String eTitle = "Zomato";
String aTitle = "";
// launch Chrome and redirect it to the Base URL
driver.get("https://www.zomato.com/");
//Maximizes the browser window
driver.manage().window().maximize();
//get the actual value of the title
aTitle = driver.getTitle();
//compare the actual title with the expected title
if (aTitle.equals(eTitle)) {
System.out.println("Test Passed");
} else {
System.out.println("Test Failed");
}
driver.quit();
//close browser
//driver.close();
}
}
```

OUTPUT: -





If incorrect title provided test is failed



EXPLICIT WAIT

Steps: -

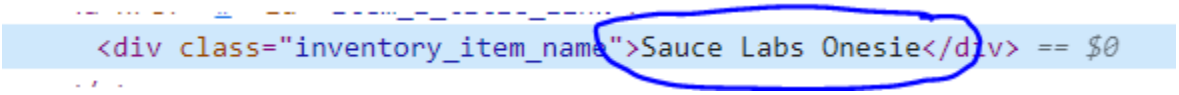
1) So first we need to give the title and the url.

2) So we have to do Ctrl+Shift+I by clicking the selecting icon hover on the product and select to find it's details.

- By.name() → Here we have to give name of component
- By.id() → Here we have to give id of component.
- sendKeys("") → Here in place of quotes value which we want to pass example username or password

3) webElement = wait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Sauce Labs Onesie")));

So in above part where “Sauce Labs Onesie” is there it is the text which is shown

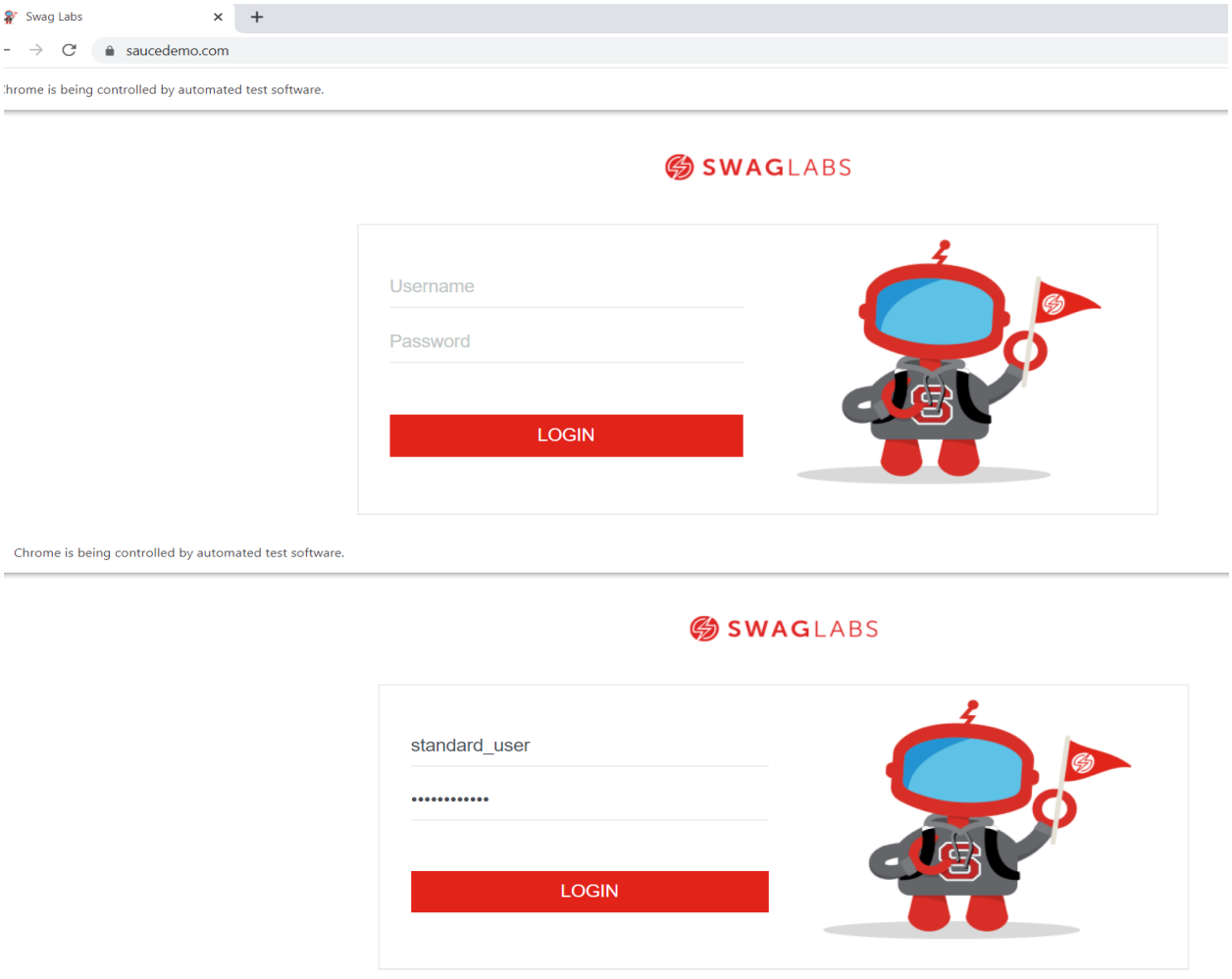


CODE: -

```
package stqa7b;
import java.time.Duration;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
/**
 *
 * @author Philip
 */
public class Stqa7b {
 static WebDriver driver;
public static void main(String[] args) throws InterruptedException {
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");
driver = new ChromeDriver();
// explicit wait - to wait for the compose button to be click-able
Duration durationInMinutes = Duration.ofMinutes(30);
WebDriverWait wait = new WebDriverWait(driver, durationInMinutes);
// launch Chrome and redirect it to the Base URL
String eTitle = "Swag Labs";
String aTitle = "";
// launch Chrome and redirect it to the Base URL
driver.get("https://www.saucedemo.com/");
//Maximizes the browser window
driver.manage().window().maximize();
//get the actual value of the title
aTitle = driver.getTitle();
//compare the actual title with the expected title
if (aTitle.contentEquals(eTitle)) {
System.out.println("Test Passed");
```

```
} else {
System.out.println("Test Failed");
}
driver.findElement(By.name("user-name")).sendKeys("standard_user");
Thread.sleep(2000);
driver.findElement(By.id("password")).sendKeys("secret_sauce");
Thread.sleep(2000);
driver.findElement(By.id("login-button")).click();
Thread.sleep(2000);
WebElement webElement;
webElement = wait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Sauce Labs Onesie")));
webElement.click();
}
}
```

OUTPUT: -



Chrome is being controlled by automated test software.

SWAGLABS

PRODUCTS

Sauce Labs Backpack

carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.

$29.99

ADD TO CART

Sauce Labs Bike Light

A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.

$9.99

Sauce Labs Bolt T-Shirt

Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun

Sauce Labs Fleece Jacke

It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling

Chrome is being controlled by automated test software.

SWAGLABS

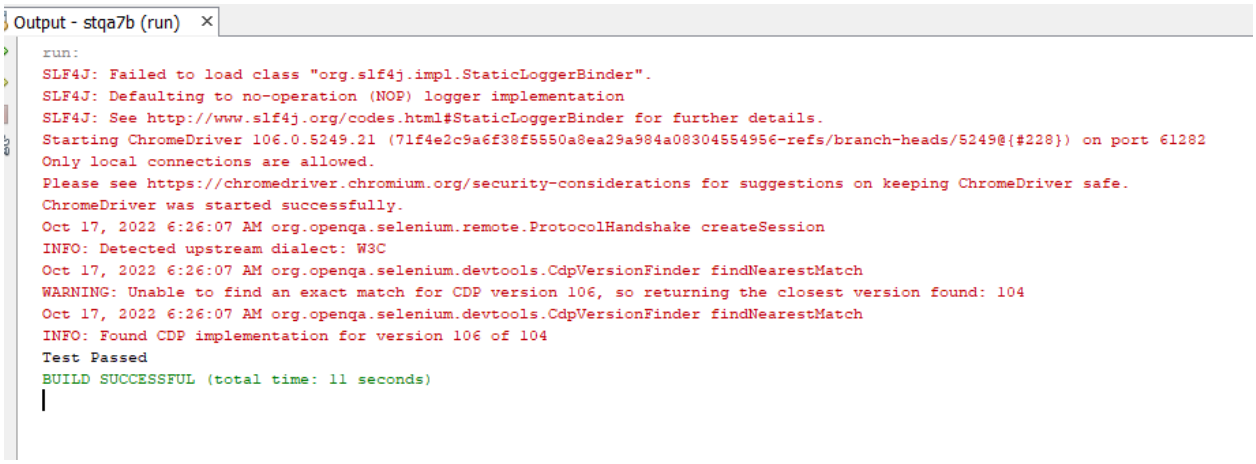
◄ BACK TO PRODUCTS

Sauce Labs Onesie

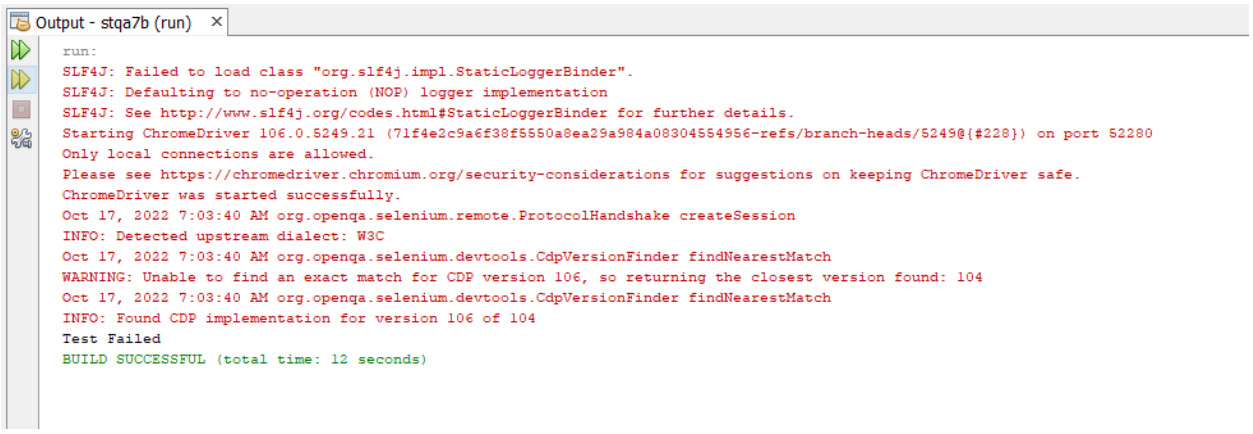
Rib snap infant onesie for the junior automation engineer in development. Reinforced 3-snap bottom closure, two-needle hemmed sleeved and bottom won't unravel.

$7.99

ADD TO CART



If incorrect title provided test is failed



CONCLUSION: -

From this practical I have learned to implement Implicit and Explicit wait.

PRACTICAL 8

AIM: - Demonstrate different types of alerts

THEORY: -

What is Alert in Selenium?

An Alert in Selenium is a small message box which appears on screen to give the user some information or notification. It notifies the user with some specific information or error, asks for permission to perform certain tasks and it also provides warning messages as well.

How to Handle Alert in Selenium WebDriver:

1) To click on the 'Cancel' button of the alert.

void dismiss()

driver.switchTo().alert().dismiss();

2) To click on the 'OK' button of the alert.

void accept()

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

3) To capture the alert message.

String getText()

driver.switchTo().alert().getText();

4) To send some data to the alert box.

void sendKeys(String stringToSend)

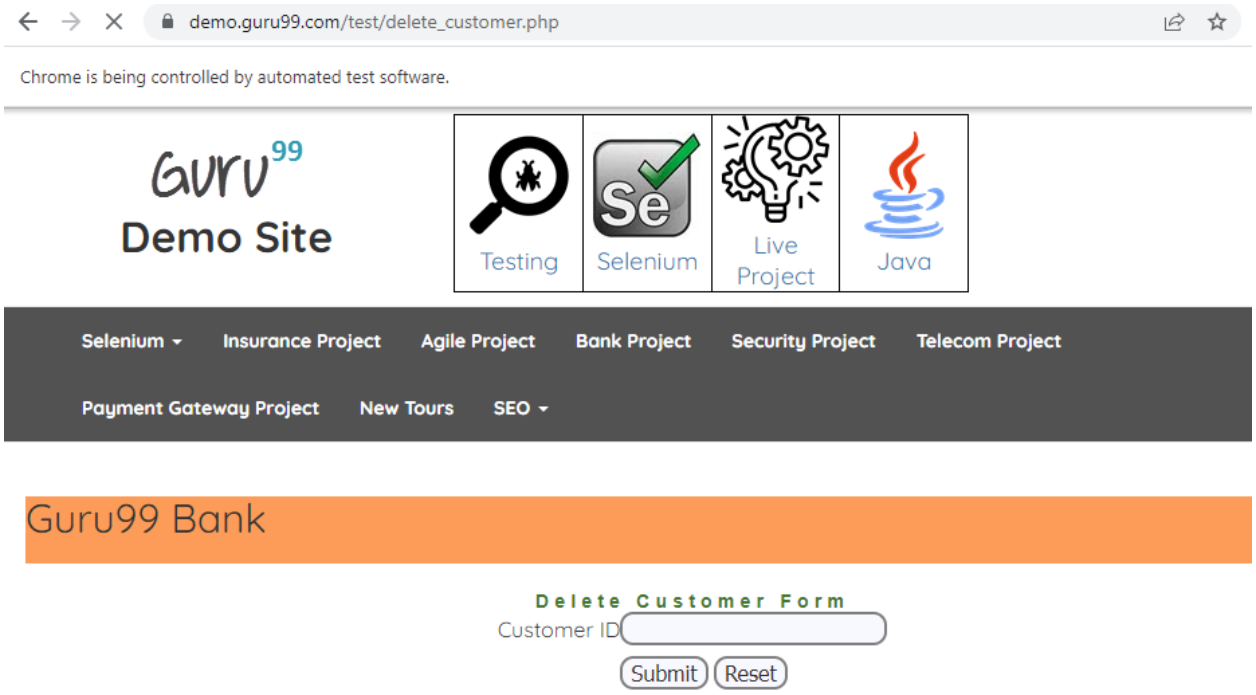
driver.switchTo().alert().sendKeys("Text");

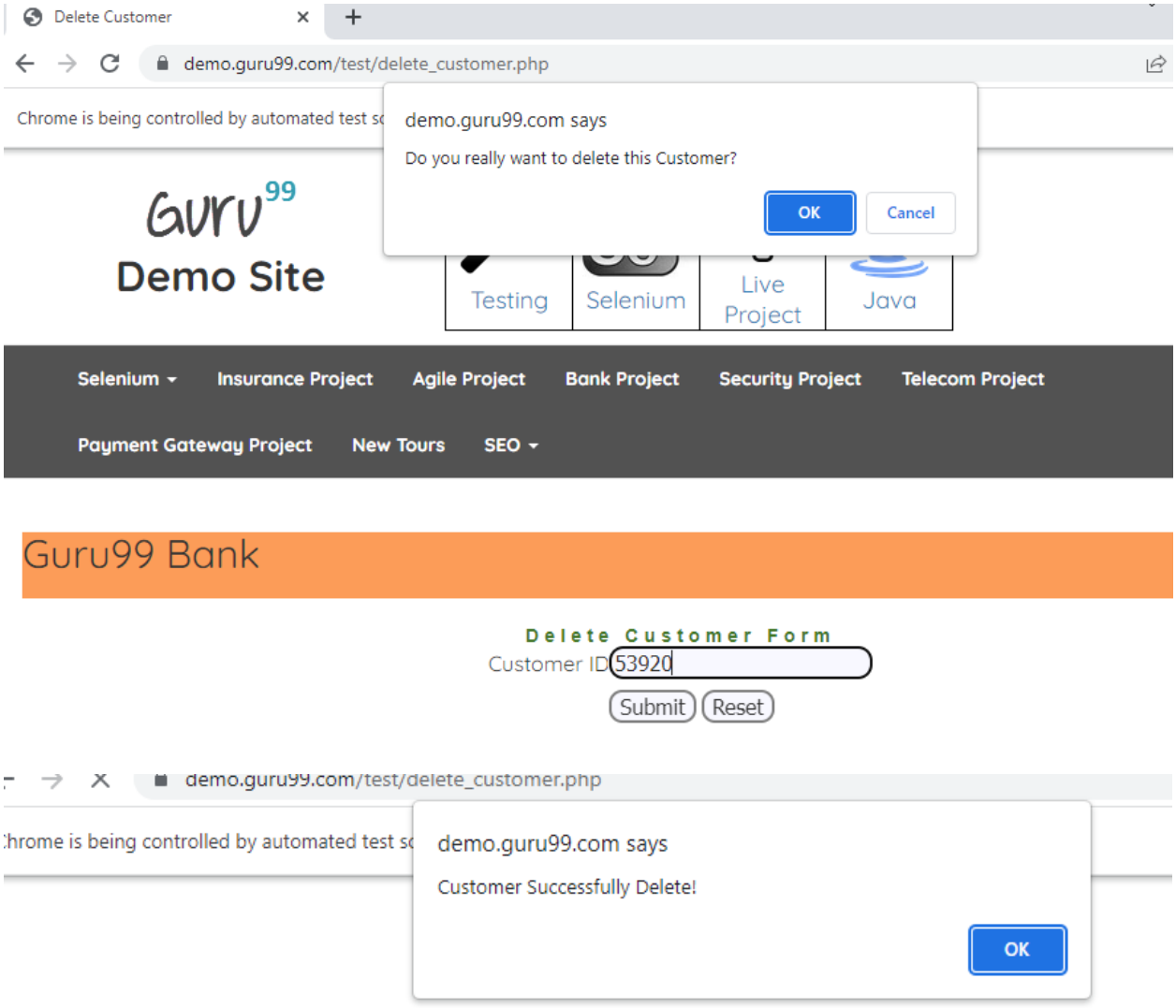
CODE: -

```
package stqa8;
import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
/**
 *
 * @author STUDENT
 */
public class Stqa8 {
```

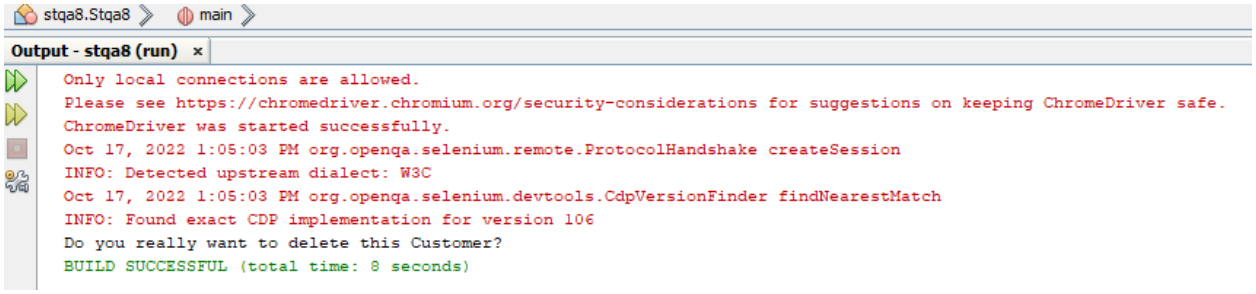
```
static WebDriver wd;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver", "E:\\CHRISTY\\CHROME\\chromedriver.exe");
wd = new ChromeDriver();
wd.get("http://demo.guru99.com/test/delete_customer.php");
wd.findElement(By.name("cusid")).sendKeys("53920");
wd.findElement(By.name("submit")).click();
// Switching to Alert
Alert alert = wd.switchTo().alert();
// Capturing alert message.
String alertMessage = wd.switchTo().alert().getText();
// Displaying alert message
System.out.println(alertMessage);
Thread.sleep(2000);
}
}
```

OUTPUT: -




Delete Customer
demo.guru99.com/test/delete_customer.php
Chrome is being controlled by automated test so
demo.guru99.com says
Do you really want to delete this Customer?
OK
Cancel
Guru99
Demo Site
Testing
Selenium
Live Project
Java
Selenium
Insurance Project
Agile Project
Bank Project
Security Project
Telecom Project
Payment Gateway Project
New Tours
SEO
Guru99 Bank
Delete Customer Form
Customer ID 53920
Submit
Reset

demo.guru99.com/test/delete_customer.php
Chrome is being controlled by automated test so
demo.guru99.com says
Customer Successfully Delete!
OK



CONCLUSION: -

From this practical I have learned to implement demonstration of different types of alerts.

PRACTICAL 9

AIM: - Demonstrate Handling Drop Down and List Boxes.

THEORY: -

Select Class in Selenium

The Select Class in Selenium is a method used to implement the HTML SELECT tag. The html select tag provides helper methods to select and deselect the elements. The Select class is an ordinary class so New keyword is used to create its object and it specifies the web element location.

Select Option from Drop-Down Box

Following is a step-by-step process on how to select value from dropdown in Selenium:

Before handling dropdown in Selenium and controlling drop-down boxes, we must do following two things:

1. Import the package **org.openqa.selenium.support.ui.Select**
2. Instantiate the drop-down box as an object, Select in Selenium WebDriver

ListBox:

ListBox is an element where user can select/deselect one or more items from it.

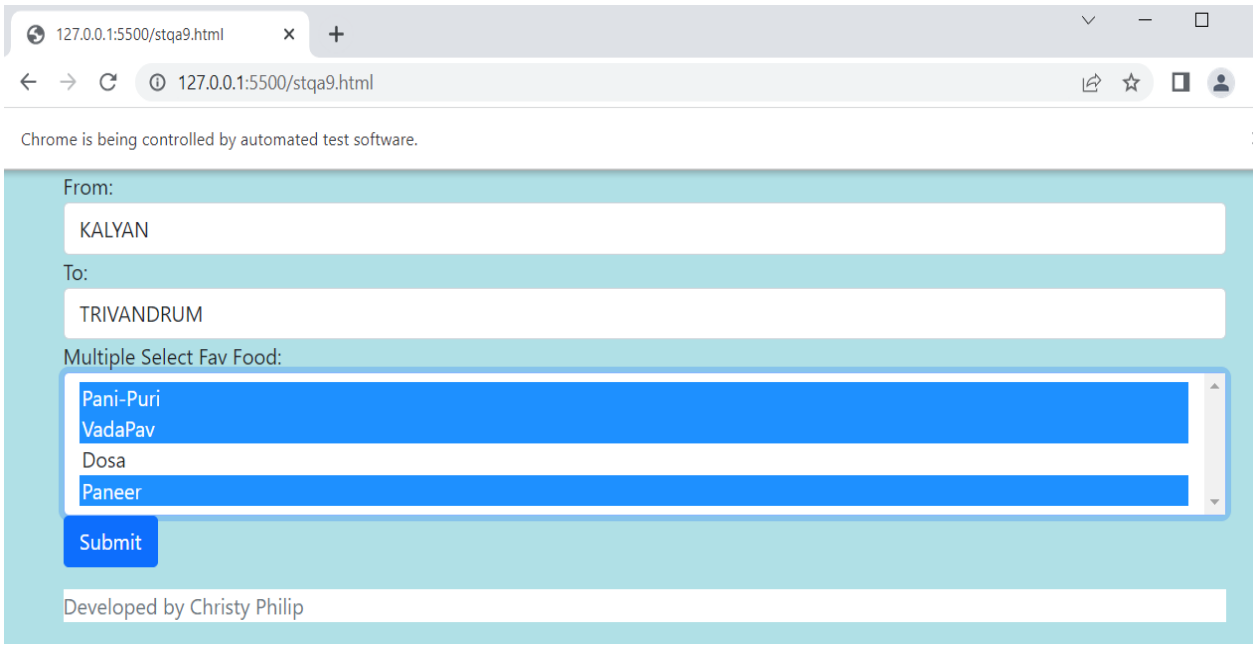
To identify the ListBox on webpage, look for select tag and attribute should be ‘multiple’ to select multiple items and there will be option tag which contains each item in it.

CODE: -

```
package stqa9;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
/**
 *
```

```
 * @author Philip
 */
public class Stqa9 {
static WebDriver wd;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver","D:\\WEB_BROWSER_DRIVER\\chromedriver
_win32\\chromedriver.exe");
wd = new ChromeDriver();
wd.get("http://127.0.0.1:5500/stqa9.html");
Select s = new Select(wd.findElement(By.name("fromStn")));
Select t = new Select(wd.findElement(By.name("toStn")));
s.selectByVisibleText("KALYAN");
t.selectByVisibleText("TRIVANDRUM");
String sMessage = s.getFirstSelectedOption().getText();
String tMessage = t.getFirstSelectedOption().getText();
System.out.println(sMessage);
System.out.println(tMessage);
Thread.sleep(2000);
Select list = new Select(wd.findElement(By.name("favFood")));
if (list.isMultiple()) {
list.selectByIndex(0);
list.selectByValue("vadavpav");
list.selectByVisibleText("Paneer");
System.out.println("Selected:");
List<WebElement> selected1 = list.getAllSelectedOptions();
for (WebElement el : selected1) {
System.out.println("Selected: " + el.getAttribute("value"));
}
Thread.sleep(5000);
list.deselectByIndex(3);
list.deselectAll();
System.out.println("Deselected");
}
}
}
```

OUTPUT: -




```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 106.0.5249.21 (71f4e2c9a6f38f5550a8ea29a984a08304554956-refs/branch-heads/5249@{#228}) on port 62801
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Oct 30, 2022 11:25:57 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 30, 2022 11:25:57 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 30, 2022 11:25:57 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
KALYAN
TRIVANDRUM
Selected:
Selected: panipuri
Selected: vadavpav
Selected: paneer
Deselected
BUILD SUCCESSFUL (total time: 14 seconds)
```

```
KALYAN
TRIVANDRUM
Selected:
Selected: panipuri
Selected: vadavpav
Selected: paneer
Deselected
BUILD SUCCESSFUL (total time: 14 seconds)
```

CONCLUSION: -

From this practical I have learned to demonstrate handling Drop Down and List Boxes.

PRACTICAL 10

AIM: - Demonstrate Command Button, Radio buttons & text boxes

THEORY: -

Selenium WebDriver - Navigation Commands

WebDriver provides some basic Browser Navigation Commands that allows the browser to move backwards or forwards in the browser's history.

Just like the browser methods provided by WebDriver, we can also access the navigation methods provided by WebDriver by typing driver.navigate() in the Eclipse panel.

1. Navigate To Command

Method: to(String arg0) : void

In WebDriver, this method loads a new web page in the existing browser window. It accepts String as parameter and returns void.

The respective command to load/navigate a new web page can be written as:

driver.navigate().to("www.google.com");

2. Forward Command

Method: to(String arg0) : void

In WebDriver, this method enables the web browser to click on the forward button in the existing browser window. It neither accepts anything nor returns anything.

Implementation:

driver.navigate().forward();

3. Back Command:

Method: back() : void

In WebDriver, this method enables the web browser to click on the back button in the existing browser window. It neither accepts anything nor returns anything.

The respective command that takes you back by one page on the browser's history can be written as: driver.navigate().back();

4. Refresh Command:

Method: refresh() : void

In WebDriver, this method refresh/reloads the current web page in the existing browser window. It neither accepts anything nor returns anything.

The respective command that takes you back by one page on the browser's history can be written as: driver.navigate().refresh();

Radio Button

Radio Buttons too can be toggled on by using the click() method.

Input Box

Input boxes refer to either of these two types:

1) Text Fields–

Selenium input text boxes that accept typed values and show them as they are.

2) Password Fields–

Text boxes that accept typed values but mask them as a series of special characters (commonly dots and asterisks) to avoid sensitive values to be displayed

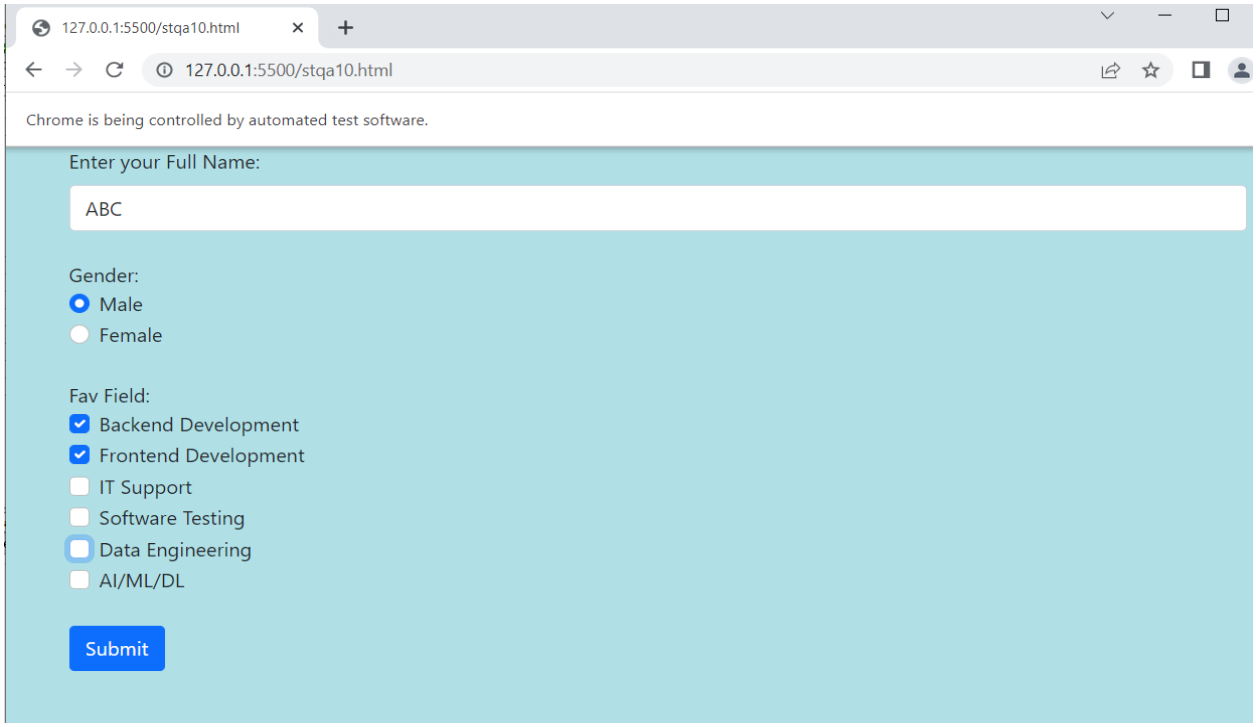
CODE: -

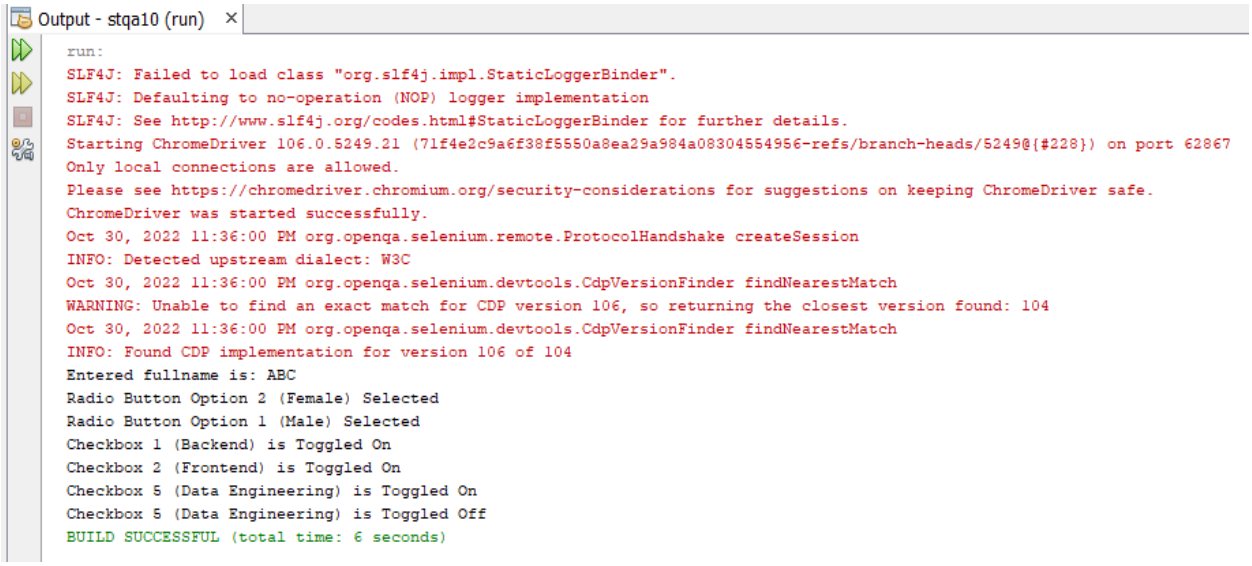
```
package stqa10;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;
/**
 *
 * @author Philip
 */
public class Stqa10 {

    /**
     * @param args the command line arguments
     */
   static WebDriver wd;
public static void main(String[] args) {
// TODO code application logic here
```

```
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");
wd = new ChromeDriver();
wd.get("http://127.0.0.1:5500/stqa10.html");
WebElement l = wd.findElement(By.name("fullname"));
l.sendKeys("ABC");
String val = l.getAttribute("value");
System.out.println("Entered fullname is: " + val);
WebElement radio1 = wd.findElement(By.id("flexRadioDefault1"));
WebElement radio2 = wd.findElement(By.id("flexRadioDefault2"));
//Radio Button1 is de-selected and Radio Button2 is selected
   radio2.click();
   System.out.println("Radio Button Option 2 (Female) Selected");
//Radio Button2 is de-selected and Radio Button1 is selected
radio1.click();
System.out.println("Radio Button Option 1 (Male) Selected");
// Selecting CheckBox
WebElement option1 = wd.findElement(By.id("flexCheckDefault4"));
WebElement option2 = wd.findElement(By.id("flexCheckDefault5"));
WebElement option3 = wd.findElement(By.id("flexCheckDefault6"));
WebElement option4 = wd.findElement(By.id("flexCheckDefault7"));
WebElement option5 = wd.findElement(By.id("flexCheckDefault8"));
// This will Toggle the Check box
option1.click();
// Check whether the Check box is toggled on
if (option1.isSelected()) {
System.out.println("Checkbox 1 (Backend) is Toggled On");
} else {
System.out.println("Checkbox 1 (Backend) is Toggled Off");
}
option2.click();
if (option2.isSelected()) {
System.out.println("Checkbox 2 (Frontend) is Toggled On");
} else {
System.out.println("Checkbox 2 (Frontend) is Toggled Off");
}
option5.click();
if (option5.isSelected()) {
System.out.println("Checkbox 5 (Data Engineering) is Toggled On");
} else {
```

```
System.out.println("Checkbox 5 (Data Engineering) is Toggled Off");
}
option5.click();
if (option5.isSelected()) {
System.out.println("Checkbox 5 (Data Engineering) is Toggled On");
}
else {
System.out.println("Checkbox 5 (Data Engineering) is Toggled Off");
}
}
}
```

OUTPUT: -






```
Entered fullname is: ABC
Radio Button Option 2 (Female) Selected
Radio Button Option 1 (Male) Selected
Checkbox 1 (Backend) is Toggled On
Checkbox 2 (Frontend) is Toggled On
Checkbox 5 (Data Engineering) is Toggled On
Checkbox 5 (Data Engineering) is Toggled Off
BUILD SUCCESSFUL (total time: 6 seconds)
```


CONCLUSION: -

From this practical I have learned to demonstrate Command Button, Radio buttons & text boxes

PRACTICAL 11

AIM: - Demonstrate action classes in Selenium

THEORY: -

What is Action Class in Selenium?

Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop, clicking on multiple elements with the control key, among others.

These operations are performed using the advanced user interactions API. It mainly consists of Actions that are needed while performing these operations.

Action class is defined and invoked using the following syntax:

```
Actions action = new Actions(driver);
action.moveToElement(element).click().perform();
```

Methods of Action Class:

Action class is useful mainly for mouse and keyboard actions. In order to perform such actions, Selenium provides various methods.

Mouse Actions in Selenium:

- doubleClick(): Performs double click on the element
- clickAndHold(): Performs long click on the mouse without releasing it
- dragAndDrop(): Drags the element from one point and drops to another
- moveToElement(): Shifts the mouse pointer to the center of the element
- contextClick(): Performs right-click on the mouse

Keyboard Actions in Selenium:

- sendKeys(): Sends a series of keys to the element
- keyUp(): Performs key release
- keyDown(): Performs keypress without release

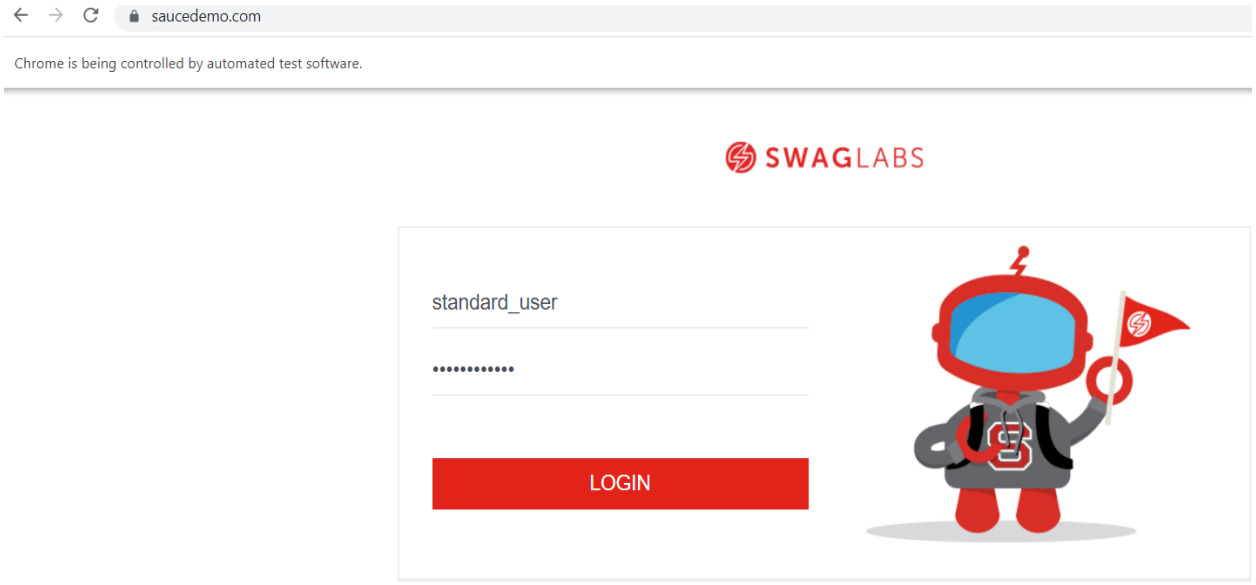
CODE: -

```
package stqa11;
import java.util.List;
import org.openqa.selenium.By;
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.support.ui.ExpectedConditions;
/**
 *
 * @author Philip
 */
public class Stqa11 {
static WebDriver wd;
public static void main(String[] args) throws InterruptedException {
// TODO code application logic here
System.setProperty("webdriver.chrome.driver",
"D:\\WEB_BROWSER_DRIVER\\chromedriver_win32\\chromedriver.exe");
wd = new ChromeDriver();
wd.get("https://www.saucedemo.com/");
wd.manage().window().maximize();
wd.findElement(By.name("user-name")).sendKeys("standard_user");
Thread.sleep(2000);
wd.findElement(By.id("password")).sendKeys("secret_sauce");
Thread.sleep(2000);
wd.findElement(By.id("login-button")).click();
Thread.sleep(2000);
System.out.println("Logged in");
Actions act = new Actions(wd);
List<WebElement> menu =
wd.findElements(By.xpath("/html/body/div/div/div/div[2]/div/div"));
System.out.println("Menu List");
for (int i = 0; i <= menu.size() - 1; i++) {
System.out.println(menu.get(i).getText() + "\n");
//print text of all the element on console
act.moveToElement(menu.get(i)).click();
//to perform mousehover on all elements of list
Thread.sleep(2000);
}
```

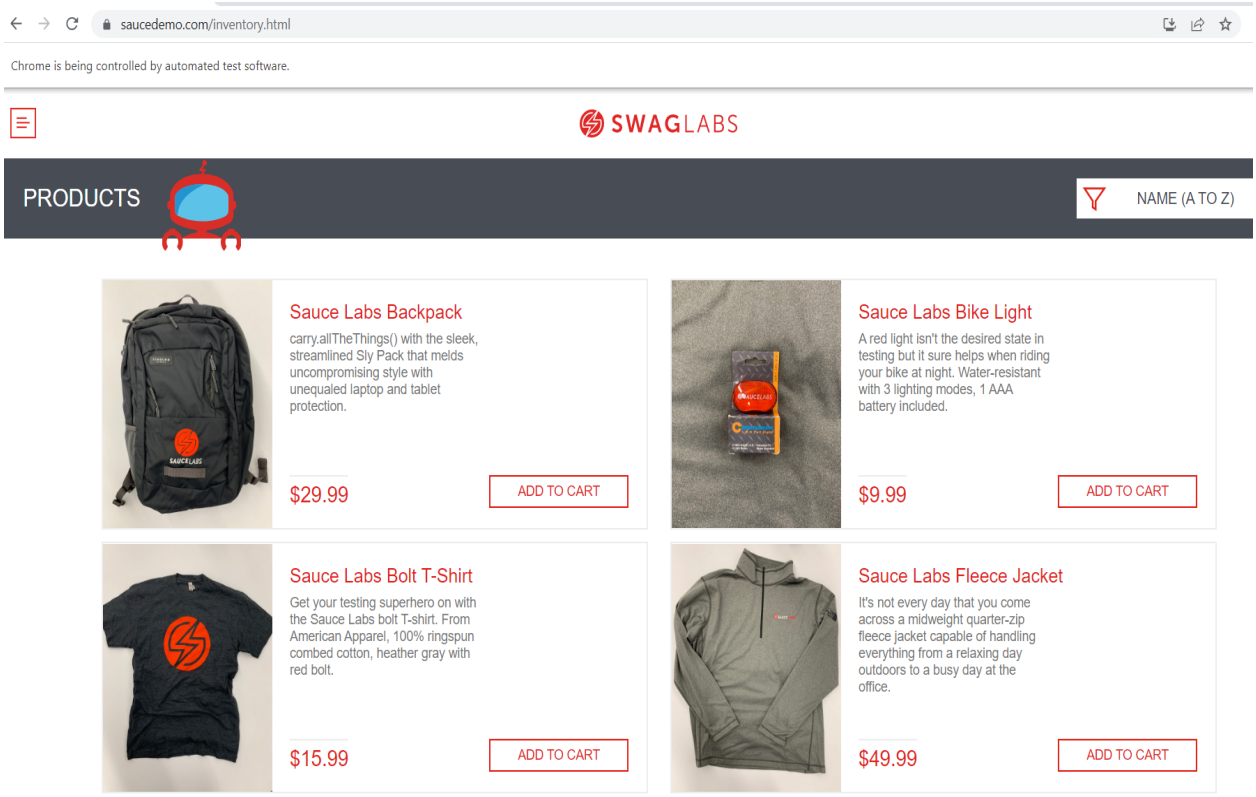
```
wd.navigate().to("https://saucelabs.com/");
Thread.sleep(2000);
System.out.println("Logout");
wd.close();
}
}
```

OUTPUT: -

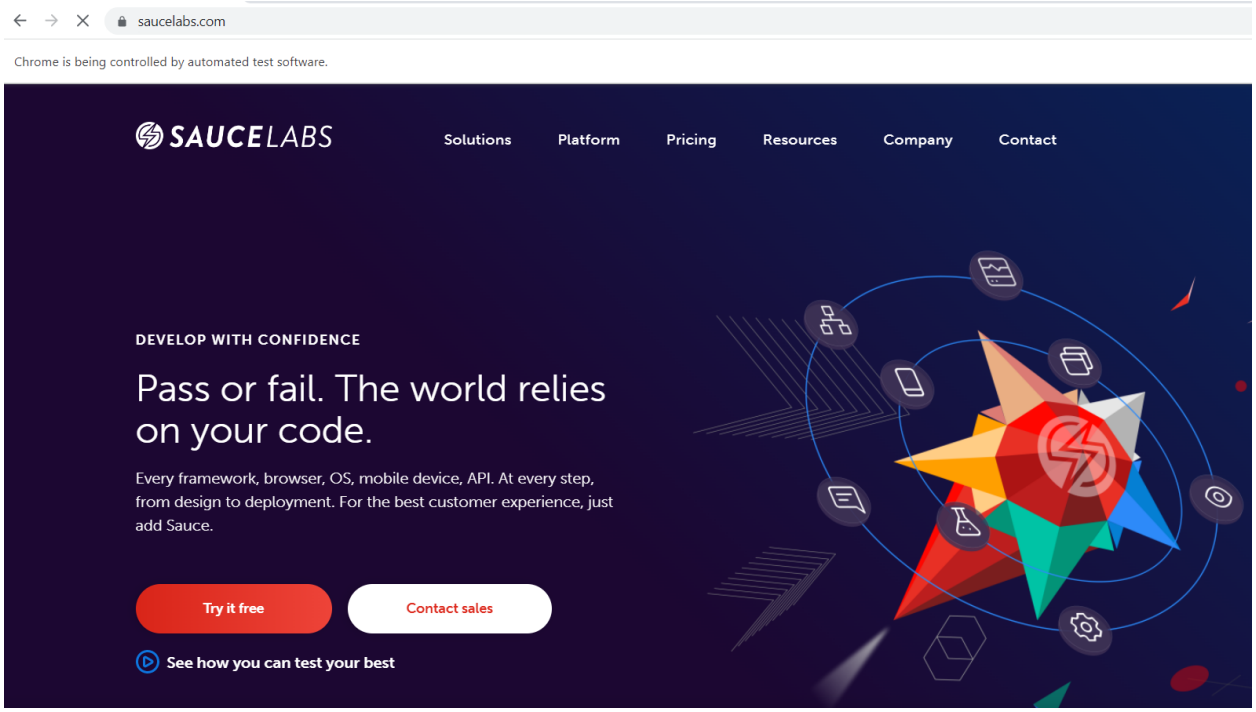
Login:



Get Menu List & Mouse Hover:



Redirecting to the official page:



Printing the Logs:

```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 106.0.5249.21 (71f4e2c9a6f38f5550a8ea29a984a08304554956-refs/branch-heads/5249@{#228}) on port 52877
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1668333060.470][WARNING]: This version of ChromeDriver has not been tested with Chrome version 107.
Nov 13, 2022 3:21:00 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Nov 13, 2022 3:21:00 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 107, so returning the closest version found: 104
Nov 13, 2022 3:21:00 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 107 of 104
Logged in
Menu List
Sauce Labs Backpack
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.
$29.99
ADD TO CART
Sauce Labs Bike Light
A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.
$9.99

Sauce Labs Bike Light
A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.
$9.99
ADD TO CART
Sauce Labs Bolt T-Shirt
Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.
$15.99
ADD TO CART
Sauce Labs Fleece Jacket
It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.
$49.99
ADD TO CART
Sauce Labs Onesie
Rib snap infant onesie for the junior automation engineer in development. Reinforced 3-snap bottom closure, two-needle hemmed sleeved and bottom won't unravel.
$7.99
ADD TO CART
Test.allTheThings() T-Shirt (Red)
This classic Sauce Labs t-shirt is perfect to wear when cozying up to your keyboard to automate a few tests. Super-soft and comfy ringspun combed cotton.
$15.99
ADD TO CART

Logout
[1668333079.637][WARNING]: Timed out connecting to Chrome, retrying...
[1668333087.735][WARNING]: Timed out connecting to Chrome, retrying...
BUILD SUCCESSFUL (total time: 35 seconds)
```

CONCLUSION: -

From this practical I have learned to Demonstrate action classes in Selenium

PRACTICAL 12

AIM: - Installation of TestNg, running testNg and TestNg annotations.

THEORY: -

What is TestNG?

TestNG is an automation testing framework in which NG stands for "Next Generation". TestNG is inspired by JUnit which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make end-to-end testing easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

- Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method fails. Then correct the errors present in the fourth method, now you want to run only the fourth method because the first three methods are anyway executed successfully. This is not possible without using TestNG.
- The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases, that means you run this XML file. It will execute only failed test cases.

Beside the above concept, you will learn more on TestNG, like what are the Advantages of TestNG, how to create test methods using @test annotations, how to convert these classes into testing suite file and execute through the eclipse as well as from the command line.

Note: -

1) Add the plugin of TestNG

Go to Eclipse→Eclipse Marketplace(By clicking help you will find)

2) Click Install

3) When a prompt comes of trust do Select all

4) After restart will be asked after doing it.

5) In Eclipse→In file→New→Java Project→Create it

6) After creating add library by right clicking project→properties→Java Build path→choose Libraries→Add Library→TestNG

7) In Eclipse→In file→Other→TestNG(TestNgClass)

8) You are all set to go

Note: -

1) Library to be compulsory added TestNG(step 6).

2) The java file must be in src it shouldn't be inside TestNG.

CODE: -

```
import org.testng.annotations.AfterClass;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.AfterSuite;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.BeforeMethod;
import org.testng.annotations.BeforeSuite;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

public class stqa12 {
	@Test
	public void f() {
		System.out.println("Test 1");
	}

	@Test
	public void f1() {
		System.out.println("Test 2");
	}

	@BeforeMethod
	public void beforeMethod() {
		System.out.println("Before Method");
```

```
	}

	@AfterMethod
	public void afterMethod() {
		System.out.println("After Method");
	}

	@BeforeClass
	public void beforeClass() {
		System.out.println("Before Class");
	}

	@AfterClass
	public void afterClass() {
		System.out.println("After Class");
	}

	@BeforeTest
	public void beforeTest() {
		System.out.println("Before Test");
	}

	@AfterTest
	public void afterTest() {
		System.out.println("After Test");
	}

	@BeforeSuite
	public void beforeSuite() {
		System.out.println("Before Suite");
	}

	@AfterSuite
	public void afterSuite() {
		System.out.println("After Suite");
	}
}
```

OUTPUT: -

```
<terminated> stqa12 [TestNG] C:\Users\Philip\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3
[RemoteTestNG] detected TestNG version 7.4.0
Before Suite
Before Test
Before Class
Before Method
Test 1
After Method
Before Method
Test 2
After Method
After Class
After Test
PASSED: f
PASSED: f1

===============================================
    Default test
    Tests run: 2, Failures: 0, Skips: 0
===============================================

After Suite

===============================================
Default suite
Total tests run: 2, Passes: 2, Failures: 0, Skips: 0
===============================================
```

CONCLUSION: -

From this practical I have learned Installation of TestNg, running testNg and TestNg annotations.

PRACTICAL 13

AIM: - Demonstrate data driven Framework.

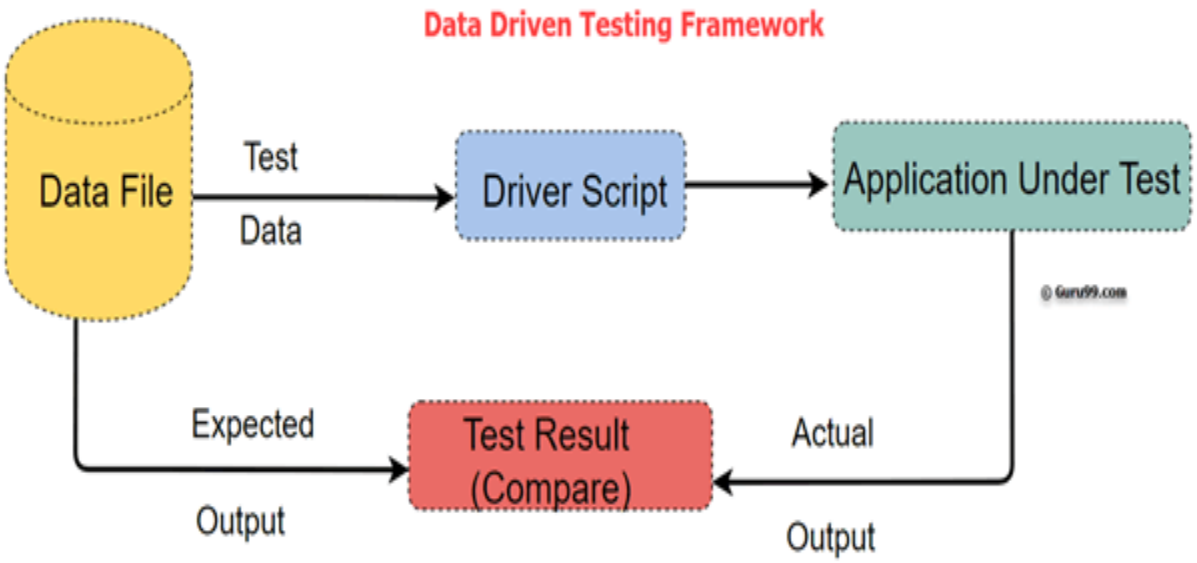
THEORY: -

Data Driven Testing:

Data Driven Testing is a software testing method in which test data is stored in table or spreadsheet format. Data driven testing allows testers to input a single test script that can execute tests for all test data from a table and expect the test output in the same table. It is also called table-driven testing or parameterized testing.

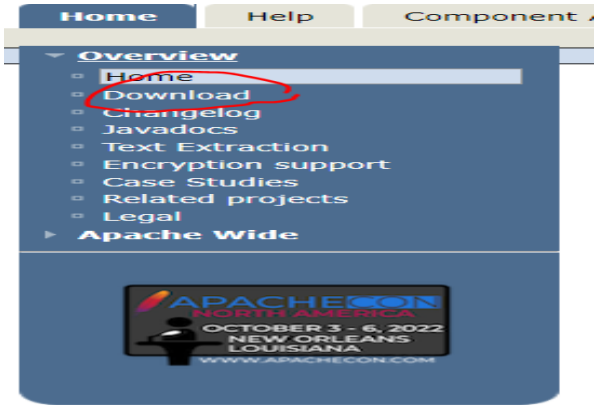
Data Driven Framework:

Data Driven Framework is an automation testing framework in which input values are read from data files and stored into variables in test scripts. It enables testers to build both positive and negative test cases into a single test. Input data in data driven framework can be stored in single or multiple data sources like .xls, .xml, .csv and databases.



Downloading jar for the Excel operation

https://poi.apache.org/



Click on Download

Refer below site for the process

https://www.toolsqa.com/blogs/download-apache-poi/

Steps: -

1) In Eclipse→In file→New→Java Project→Create it

2) After creating add library by right clicking project→properties→Java Build path→choose Libraries→Add Library→TestNG

3) In Eclipse→In file→Other→TestNG(TestNgClass)

4) We need Selenium Driver by right clicking project→properties→Java Build path→choose Libraries→Add External JARs→Add the Selenium Driver

5) We need POI jar which is shown above by right clicking project→properties→Java Build path→choose Libraries→Add External JARs→Add the POI jar(Refer above link for getting which all jars to be added)

6) Now create 2 files

- TestNG file(Practical13.java)
- Normal java file(Excel.java)

7) We need to create a excel file which is shown in OUTPUT section

8) You are now all set to execute code

Note: -

1) Both java file should be inside src.

2) TestNG library should be added

3) Both the jar files should be compulsory added.

CODE: -

Practical13.java

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
import org.testng.annotations.Test;
public class Practical13 {
	WebDriver wd;
	@Test(dataProvider = "testdata")
	public void demoClass(String username, String password) throws InterruptedException {

System.setProperty("webdriver.chrome.driver","D:\\WEB_BROWSER_DRIVER\\chromedriver
_win32\\chromedriver.exe");

	wd = new ChromeDriver();
	wd.get("https://www.saucedemo.com/");
	wd.manage().window().maximize();
	Thread.sleep(5000);
	wd.findElement(By.name("user-name")).sendKeys(username);
	Thread.sleep(2000);
	wd.findElement(By.id("password")).sendKeys(password);
	Thread.sleep(2000);
	wd.findElement(By.id("login-button")).click();
	Thread.sleep(2000);
	String                                                                    str
=wd.findElement(By.xpath("/html/body/div/div/div/div[1]/div[2]/span")).getText();

	if(str=="Products")
	{
	System.out.println("Login Successful");
	}
	else
	{
	System.out.println("Login Failed");
	}
	}
	@AfterMethod
```

```
	void ProgramTermination() {
		wd.quit();
	}
	@DataProvider(name = "testdata")
	public Object[][] testDataExample() {
	Excel configuration = new Excel("D:\\data.xlsx");
	int rows = configuration.getRowCount(0);
	Object[][] signin_credentials = new Object[rows][2];
	for (int i = 0; i < rows; i++) {
	signin_credentials[i][0] = configuration.getData(0, i, 0);
	signin_credentials[i][1] = configuration.getData(0, i, 1);
	}
	return signin_credentials;
	}

}
```

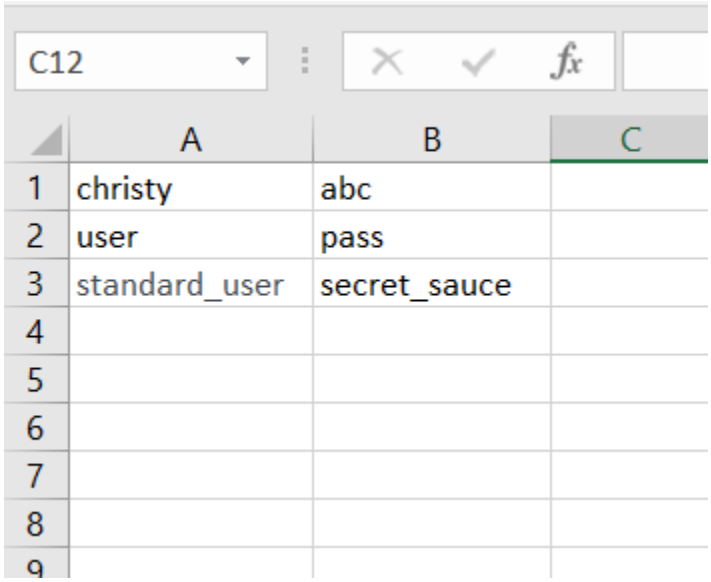
Excel.java

```
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;
public class Excel {
	XSSFWorkbook work_book;
	XSSFSheet sheet;
	public Excel(String excelfilePath) {
	try {
	File s = new File(excelfilePath);
	FileInputStream stream = new FileInputStream(s);
	work_book = new XSSFWorkbook(stream);
	} catch (Exception e) {
	System.out.println(e.getMessage());
	}
	}
	public String getData(int sheetnumber, int row, int column) {
		sheet = work_book.getSheetAt(sheetnumber);
		String data = sheet.getRow(row).getCell(column).getStringCellValue();
		return data;
		}
		public int getRowCount(int sheetIndex) {
```

```
        int row = work_book.getSheetAt(sheetIndex).getLastRowNum();
        row = row + 1;
        return row;
        }

}
```

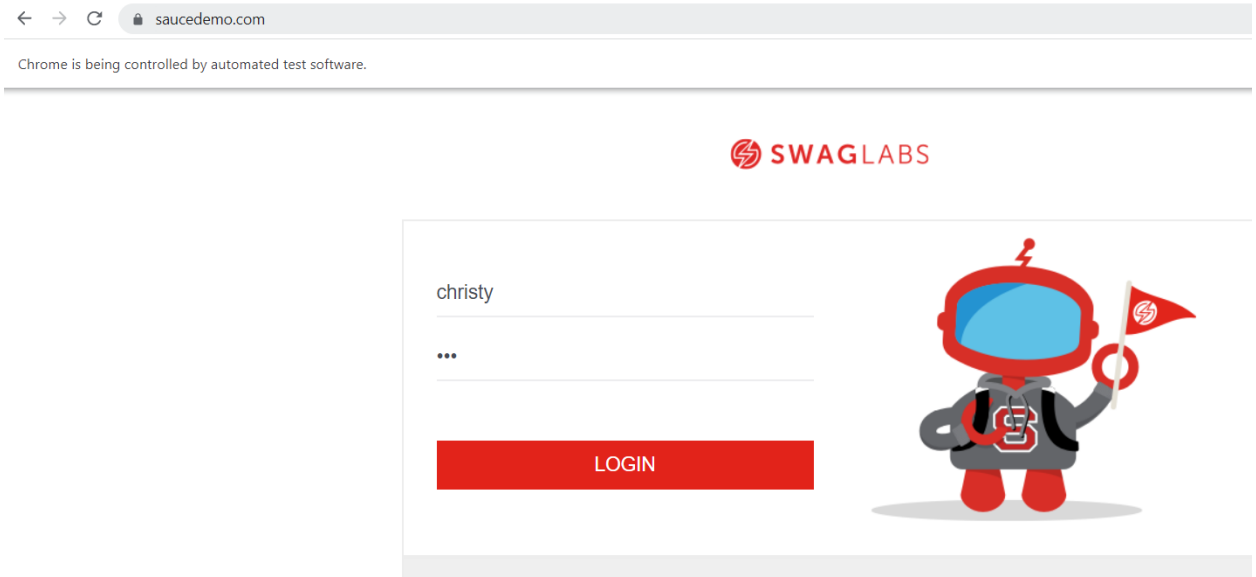
OUTPUT: -

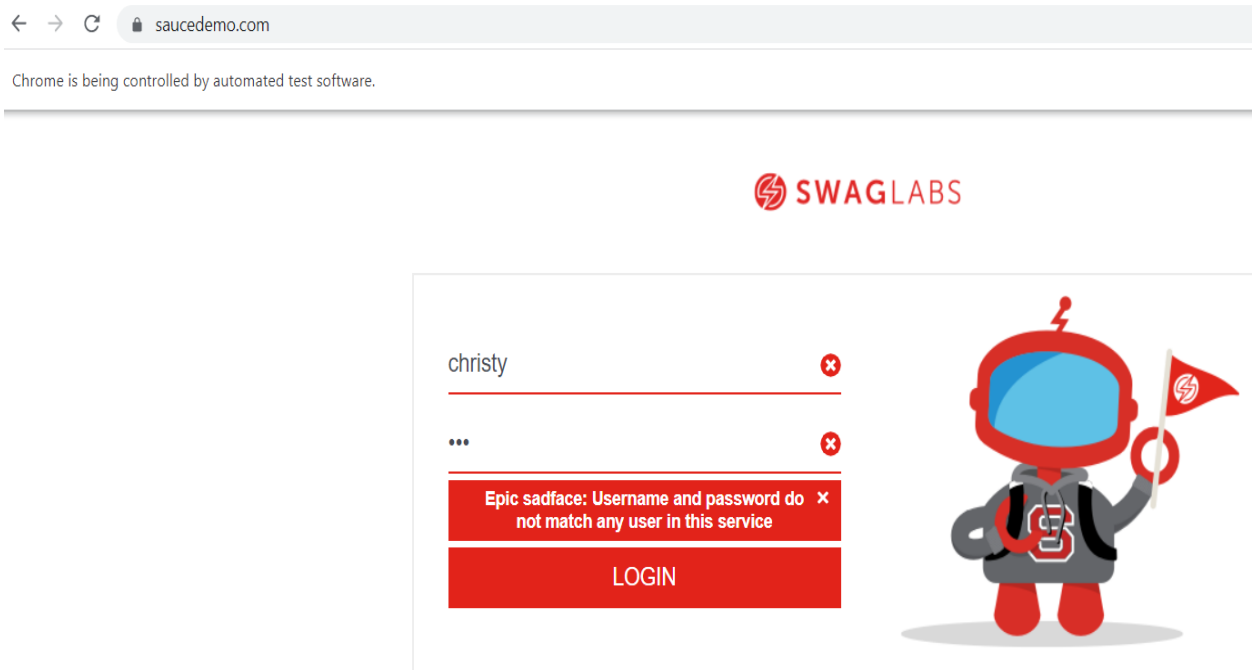
data.xlsx



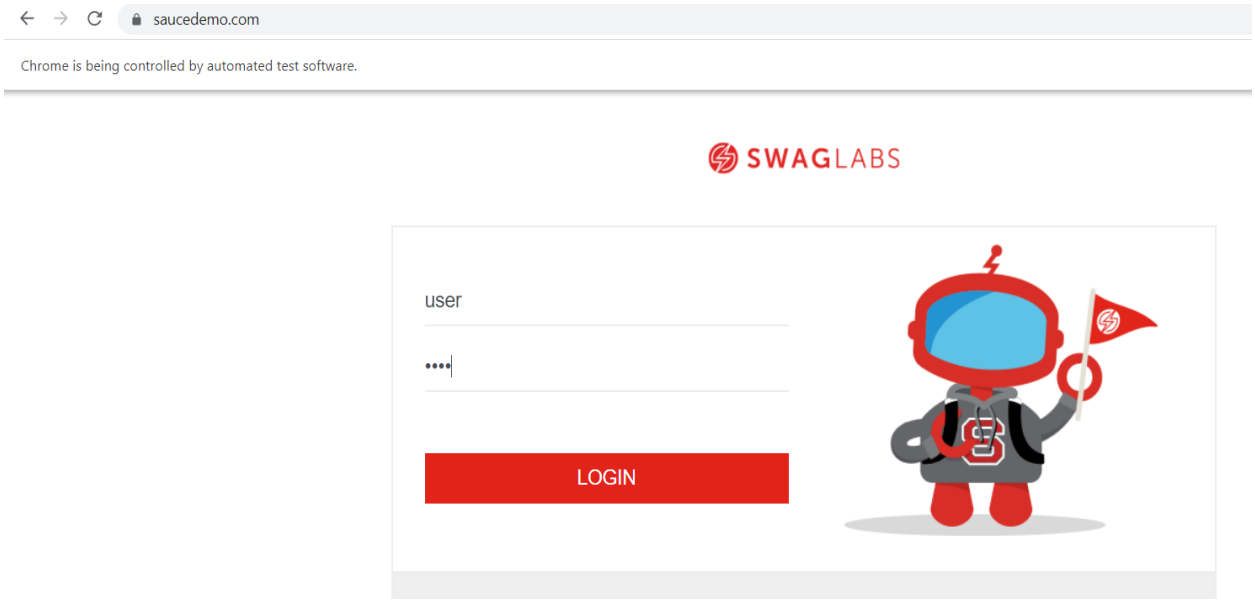
	A	B	C
1	christy	abc	
2	user	pass	
3	standard_user	secret_sauce	
4			
5			
6			
7			
8			

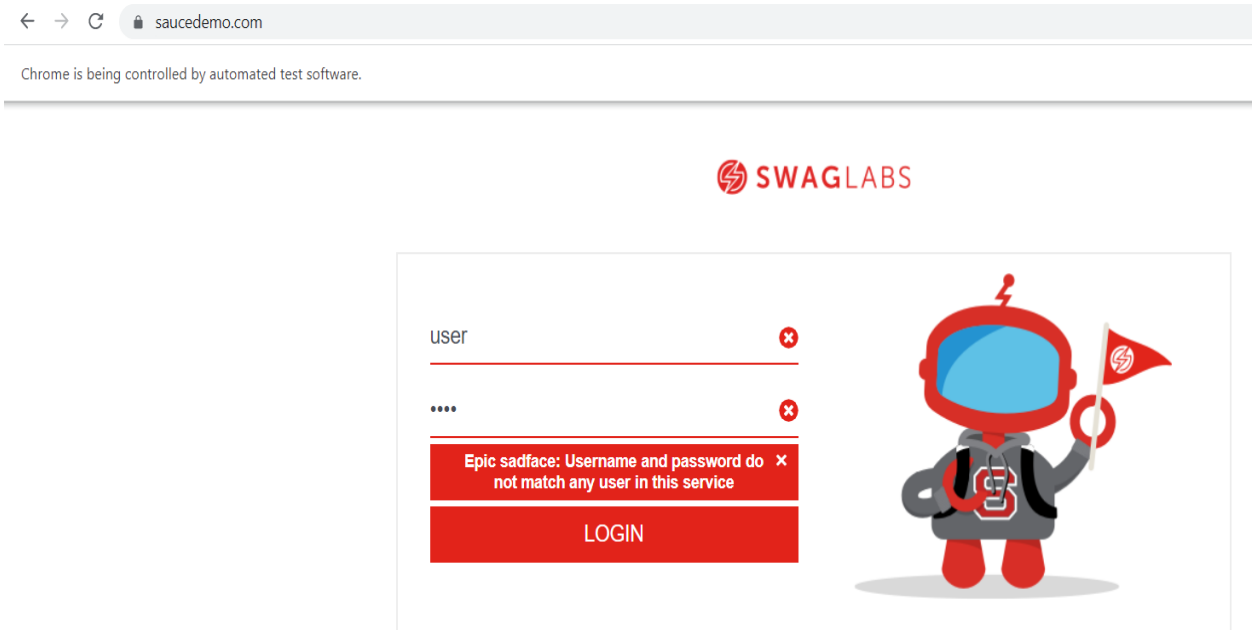
1) Try1: [failed]



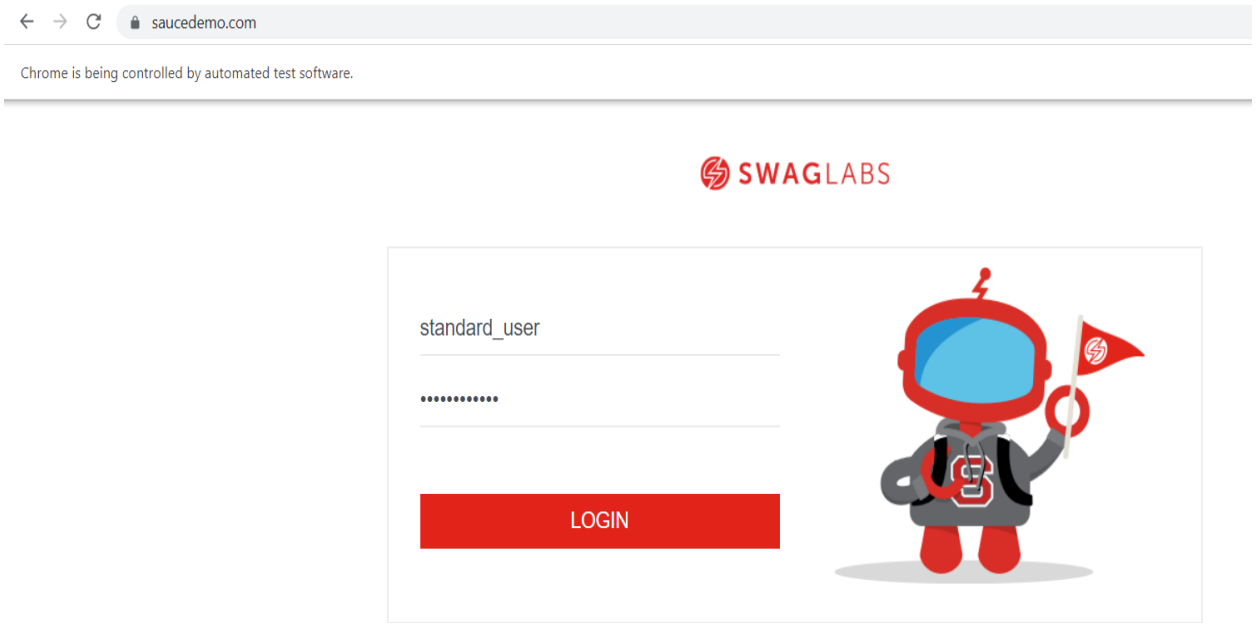


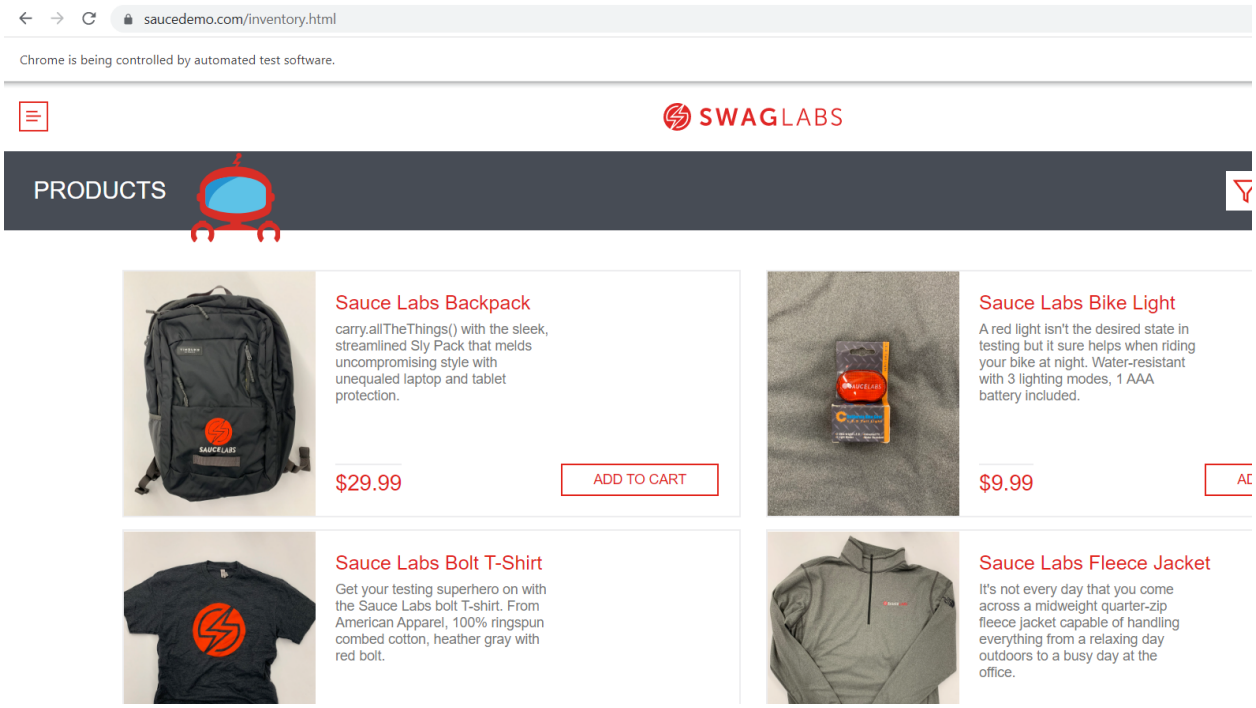
2) Try 2: [failed]





3) Try 3: [success]





Stats: Test ng:

```
===============================================
    Default test
    Tests run: 1, Failures: 2, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 3, Passes: 1, Failures: 2, Skips: 0
===============================================
```

CONCLUSION: -

From this practical I have learned to Demonstrate data driven Framework.