Roll No. 42 Exam Seat No.

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

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

Since 1962

CERTIFICATE

Certified that Mr./Miss Ronak Bipin Pathare

of SYMCA 2nd Shift

has satisfactorily completed a course of the necessary experiments in

Software Testing and Quality Assurance Lab under my supervision

in the Institute of Technology in the academic year 2021 - 2022.

Principal Head of Department

Faculty In-charge External Examiner

Index

Sr. No	Contents	Date of Preparation	Date of Submission	Page Number	Marks	Sign
1	Take a review and write testcases for any known application.	27/8/2021	3/9/2021	2		
2	Implement Web Drivers on Chrome & Firefox Browsers.	3/9/2021	9/9/2021	12		
3	Demonstrate handling multiple frames in selenium	17/9/2021	24/9/2021	15		
4	Implement Browser command and navigation command	17/9/2021	24/9/2021	19		
5	Implement the find element command	24/9/2021	1/10/2021	25		
6	Demonstrate the Locator (id, css selector,path)	24/9/2021	1/10/2021	28		
7	Demonstrate synchronization in selenium	1/10/2021	8/10/2021	32		
8	Demonstrate different types of alerts	8/10/2021	14/10/2021	38		
9	Demonstrate: Handling Drop Down,List Boxes	22/10/2021	29/10/2021	41		
10	Demonstrate: Command Button, Radio buttons & text boxes.Waits command in selenium	29/10/2021	8/11/2021	44		
11	Demonstrate action classes in Selenium	12/11/2021	18/11/2021	47		
12	Installation of TestNg, running TestNg and TestNg annotations	26/11/2021	3/12/2021	50		
13	Demonstrate data driven Framework	3/12/2021	12/12/2021	53		

Practical 1

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

Theory:

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenarios to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

Best Practice for writing good Test Case Example.

1. **Test Cases need to be simple and transparent:** Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them. Use assertive language like go to the home page, enter data, click on this and so on. This makes understanding the test steps easy and tests execution faster.
2. **Create Test Case with End User in Mind:** The goal of any software project is to create test cases that meet customer requirements and are easy to use and operate. A tester must create test cases keeping in mind the end user perspective
3. **Avoid test case repetition:** Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the precondition column
4. **Do not Assume:** Do not assume functionality and features of your software application while preparing a test case. Stick to the Specification Documents.
5. **Ensure 100% Coverage:** Make sure you write test cases to check all software requirements mentioned in the specification document. Use Traceability Matrix to ensure no functions/conditions are left untested.
6. **Test Cases must be identifiable:** Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.
7. **Implement Testing Techniques:** It's not possible to check every possible condition in your software application. Software Testing techniques help you select a few test cases with the maximum possibility of finding a defect.
   - **Boundary Value Analysis (BVA):** As the name suggests it's the technique that defines the testing of boundaries for a specified range of values.
   - **Equivalence Partition (EP):** This technique partitions the range into equal parts/groups that tend to have the same behaviour.
   - **State Transition Technique**: This method is used when software behaviour changes from one state to another following particular action.
   - **Error Guessing Technique:** This is guessing/anticipating the error that may arise while doing manual testing. This is not a formal method and takes advantages of a tester's experience with the application
8. **Self-cleaning:** The test case you create must return the Test Environment to the pretest state and should not render the test environment unusable. This is especially true for configuration testing.

Topic: Movers and Packers website.

Description:

- Movers and Packers website is more reliable and easier to use than the present system.
- Our solution targets those who do not have time to spare.
- We propose an easy way for finding transportation without any delay and inconvenience.
- Initially records containing details of customers were managed manually, but in the proposed system these records will be maintained in a database from which they can be updated.
- Proposed system calculates the pricing according to the travel distance and quantity of items in the transport to minimise the cost of service.
- Highly trained staff is available for customer service to minimise damage to goods and improve companies' productivity.

Goals:

- **Overall quality of service:-** As a mover and packer company services will be offered to corporate as well as individual customers. User expects to be offered with a great quality services from end to end e.g. quality of packing materials, packing, process of loading, unloading and so on, everything have to be exceptionally good
- **Safety of goods:** The idea of using packers and movers service is pretty simple. As a user, I want my goods to be packed and delivered timely and in safe and sound condition without any damage, theft or misplacement.
- **Punctuality:** Relocating to a new home is a complicated process and that's why professional packers and movers are hired. Professionalism is expected in any business but especially so in service business, where you don't have the monopoly.
- **Commitment to deliver on time:** For end customers, it is important to have their goods delivered on time because they have everything with you, so they can't start their life in the new house unless you deliver their household goods. For industrial and commercial shifting, their productivity and commitment is at stake, so you have to meet the delivery timeline.
- **Prompt and clear communication:** No one likes hidden agendas and hence a clear-cut communication can help you win customers.
- **Value your words:** This is important for any business, including the packers and mover's business. You can't end up quoting less for a task and then change your quote in between. Whatever you have committed, should be fulfilled to the best of your abilities.

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_01	**Test Designed by: Ronak Pathare**
Module Name: Login Page	**Test Designed date: 28-08-2021**
Test Scenario: Verify login with username and password	**Test Executed by: Ronak Pathare**
Description: Test the login page	**Test Execution date: 28-08-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Enter valid username and password	Need a username and password	Enter username, password and click on login	Valid username and password	Successful login	Successful login	Redirect to home page	Pass
2	Enter invalid username and valid password	Need a username and password	Enter username, password and click on login	Invalid username and Valid password	A message “Invalid login details”	Displayed message “Invalid login details”	Retain user on login page	Pass
3	Enter valid username and invalid password	Need a username and password	Enter username, password and click on login	Valid username and Invalid password	A message “Invalid login details”	Displayed message “Invalid login details”	Retain user on login page	Pass
4	Enter invalid username and invalid password	Need a username and password	Enter username, password and click on login	Invalid username and invalid password	A message “Invalid login details”	Displayed message “Invalid login details”	Retain user on login page	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_02	**Test Designed by: Ronak Pathare**
Module Name: Sign in Page	**Test Designed date: 29-08-2021**
Test Scenario: Register user	**Test Executed by: Ronak Pathare**
Description: Test the sign in page	**Test Execution date: 29-08-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Enter valid username & password	Need all required fields	Enter username, password & click on register button	Valid username & password	Successful login	Successful login	Redirect to home page	Pass
2	Enter invalid username & valid password	Need all required fields	Enter username, password & click on register button	Invalid username & Valid password	A message “Invalid email id”	Displayed message “Invalid email id”	Retain user on sign in page	Pass
3	Enter valid username & invalid password	Need all required fields	Enter username, password & click on register button	Valid username & Invalid password	A message “Invalid password format”	Displayed message “Invalid password format”	Retain user on sign in page	Pass
4	Enter valid username & invalid Confirm password	Need all required fields	Enter username, password & click on register button	Valid username & invalid confirm password	A message “Password and Confirm Password not matching”	Displayed “Password and Confirm Password not matching”	Retain user on sign in page	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_03	**Test Designed by: Ronak Pathare**
Module Name: Booking Page	**Test Designed date: 30-08-2021**
Test Scenario: Book transport service	**Test Executed by: Ronak Pathare**
Description: Test the booking page part 1	**Test Execution date: 30-08-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Enter valid source & destination	Need all required fields	Enter full source & destination address	Valid source & destination address	Successful selection of address	Successful selection of address	Redirect to date selection	Pass
2	Enter invalid source & destination	Need all required fields	Enter incomplete source & destination address	Invalid source & destination address	A message “Please enter full address”	Displayed message “Please enter full address”	Retain user on address selection	Pass
3	Enter date & time	Need all required fields	Enter date and time in correct format	Valid date and time	Successful selection of date	Successful selection of date	Redirect to vehicle selection	Pass
4	Enter invalid date & time	Need all required fields	Enter date and time in wrong format	Invalid date and time	A message “Please enter date and time correct format”	Displayed “Please enter date and time correct format”	Retain user date selection	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_04	**Test Designed by: Ronak Pathare**
Module Name: Booking Page	**Test Designed date: 30-08-2021**
Test Scenario: Book transport service	**Test Executed by: Ronak Pathare**
Description: Test the booking page part 2	**Test Execution date: 30-08-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Select vehicle type and material of transport	Need all required fields	Select correct vehicle type & material of transport	Valid vehicle type & material of transport	Successful selection of vehicle	Successful selection of vehicle	Redirect to confirm booking	Pass
2	Select invalid vehicle type & material of transport	Need all required fields	Select incorrect vehicle type & material of transport	Invalid vehicle type & material of transport	A message “Please select correct material and vehicle type”	Displayed message “Please select correct material and vehicle type”	Retain user on vehicle selection	Pass
3	Confirm booking	Need all above steps completed	View and confirm all details entered		Successful booking of transportation service	Successful booking of transportation service	Redirect to home page	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_05	**Test Designed by: Ronak Pathare**
Module Name: Update Page	**Test Designed date: 01-09-2021**
Test Scenario: Update previously booked transport service	**Test Executed by: Ronak Pathare**
Description: Test the update page part 1	**Test Execution date: 01-09-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Verify or correct source & destination	Need all required fields	Enter full source & destination address	Valid source & destination address	Successful selection of address	Successful selection of address	Redirect to date selection	Pass
2	Verify or correct source & destination	Need all required fields	Enter incomplete source & destination address	Invalid source & destination address	A message “Please enter full address”	Displayed message “Please enter full address”	Retain user on address selection	Pass
3	Verify or correct date & time	Need all required fields	Enter date and time in correct format	Valid date and time	Successful selection of date	Successful selection of date	Redirect to vehicle selection	Pass
4	Verify or correct invalid date & time	Need all required fields	Enter date and time in wrong format	Invalid date and time	A message “Please enter date and time correct format”	Displayed “Please enter date and time correct format”	Retain user date selection	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_06	**Test Designed by: Ronak Pathare**
Module Name: Update Page	**Test Designed date: 01-09-2021**
Test Scenario: Update previously booked transport service	**Test Executed by: Ronak Pathare**
Description: Test the update page part 2	**Test Execution date: 01-09-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Verify or correct vehicle type and material of transport	Need all required fields	Select correct vehicle type & material of transport	Valid vehicle type & material of transport	Successful selection of vehicle	Successful selection of vehicle	Redirect to confirm booking	Pass
2	Verify or correct vehicle type & material of transport	Need all required fields	Select incorrect vehicle type & material of transport	Invalid vehicle type & material of transport	A message “Please select correct material and vehicle type”	Displayed message “Please select correct material and vehicle type”	Retain user on vehicle selection	Pass
3	Confirm booking	Need all above steps completed	View and confirm all details entered		Successful booking of transportation service	Successful booking of transportation service	Redirect to home page	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_07	**Test Designed by: Ronak Pathare**
Module Name: Display Page	**Test Designed date: 02-09-2021**
Test Scenario: Display previously booked transport service	**Test Executed by: Ronak Pathare**
Description: Test the display page	**Test Execution date: 02-09-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Display previously booked service	User has a service booked	Click on display booking button		Successful display of all bookings	Successful display of all bookings		Pass
2	Click on update button	User has a service booked	Select a booking which user wants to update and click on update button	Confirm user wants to update this booking	Redirect to update booking page	Redirect to update booking page		Pass
3	Click on delete button	User has a service booked	Select a booking which user wants to delete and click on delete button	Confirm user wants to delete this booking	A message “Booking deleted successfully”	Displayed message “Booking deleted successfully”	Reload display booking page	Pass

Project Name : Movers and Packers	
Test Case	
Test Case ID: Test_08	**Test Designed by: Ronak Pathare**
Module Name: Delete Page Admin	**Test Designed date: 02-09-2021**
Test Scenario: Delete previously booked transport service by admin	**Test Executed by: Ronak Pathare**
Description: Test the delete page for admin	**Test Execution date: 02-09-2021**

No.	Test case	Precondition	Test Steps	Test Data	Expected Result	Actual Result	Post condition	Status
1	Enter username and date of transport	Need all required fields	Enter username & date of transport	Valid username & date of transport	Successful deletion of booking for that user & message conveying same	Successful deletion of booking for that user & message conveying same		Pass
2	Enter username and date of transport	Need all required fields	Enter username & wrong date of transport	Valid username & invalid date of transport	A message “No service of this user”	Display message “No service of this user”	Retain admin to delete booking page	Pass

Conclusion:

Various tests are successfully performed on website Movers and Packers.

Practical 2

Aim: Implement Web Drivers on Chrome & Firefox Browsers

Theory:

WebDriver

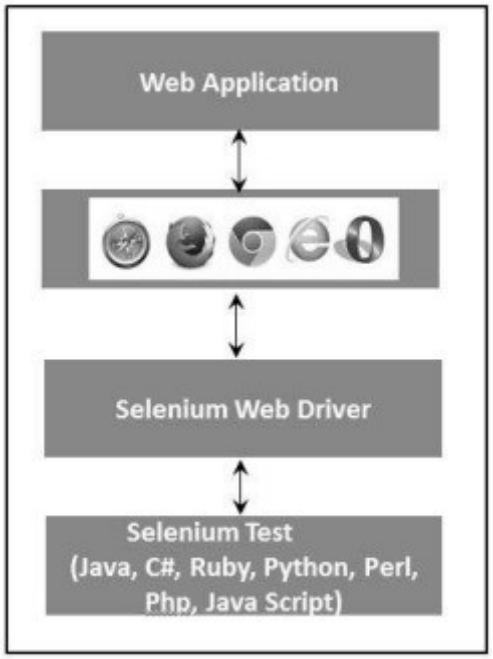
WebDriver is a remote-control interface that enables introspection and control of user agents. It provides a platform- and language-neutral wire protocol as a way for out-of-process programs to remotely instruct the behaviour of web browsers.

Provided is a set of interfaces to discover and manipulate DOM elements in web documents and to control the behaviour of a user agent. It is primarily intended to allow web authors to write tests that automate a user agent from a separate controlling process but may also be used in such a way as to allow in-browser scripts to control a possibly separate browser.

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 got 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.

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

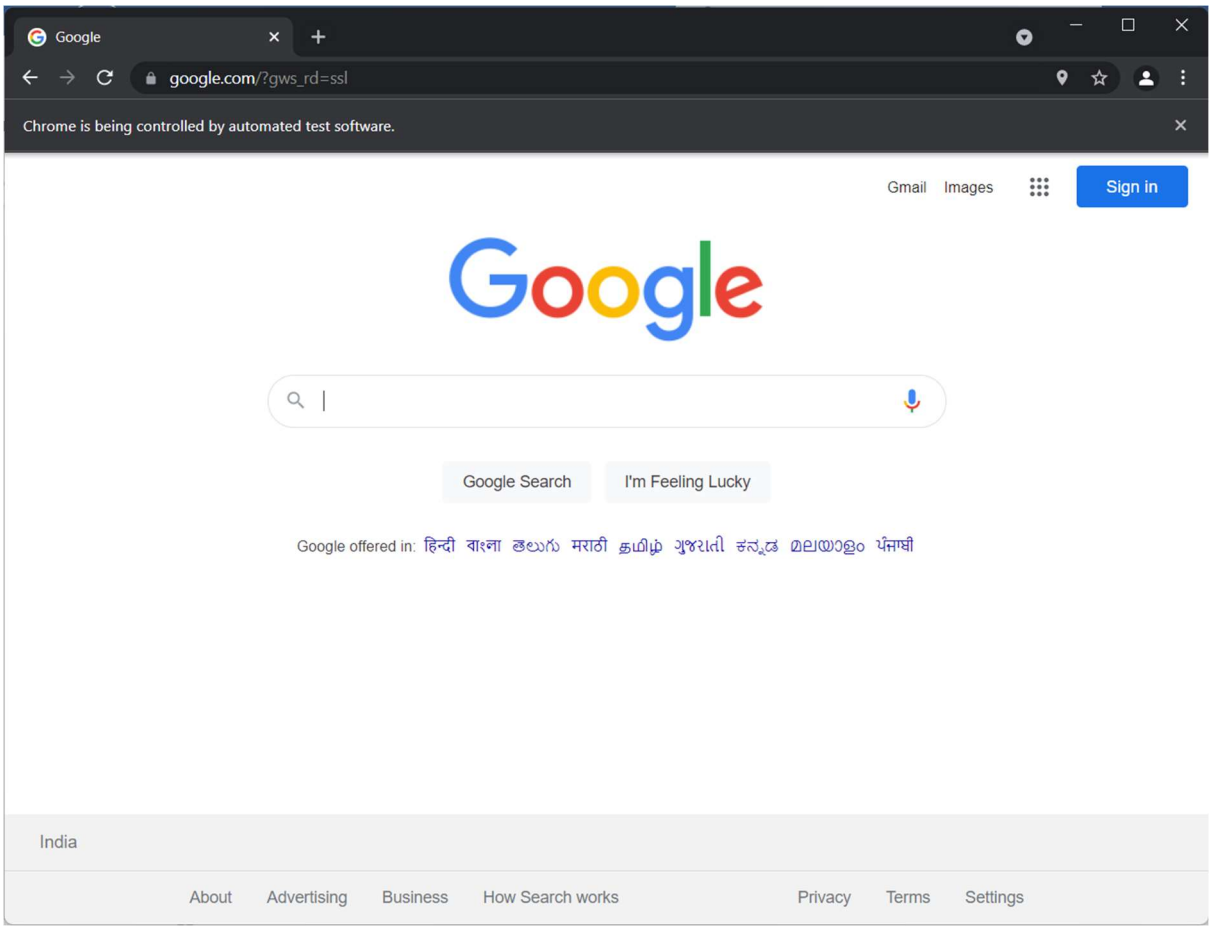


Code:

```
import java.util.Scanner;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

public class stqa02 {
	static WebDriver wd ;
	public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		Scanner sc = new Scanner(System.in);
		System.out.println("1.ChromeBrowser \n 2. Firefox Broswer");
		System.out.println("Choice");
		int ch = sc.nextInt();
		sc.close();
		switch (ch) {
		case 1:

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
			wd = new ChromeDriver();
			break;
		case 2:

	System.setProperty("webdriver.gecko.driver","C:\\Users\\bipin\\Documents\\M
CA\\SEM 3\\STQA\\geckodriver-v0.30.0-win64\\geckodriver.exe");
			wd = new FirefoxDriver();
		default:
			System.out.println("Invalid Choice");
			break;
		}
		if(wd!=null){
			wd.get("http:\\google.com");
		}
	}

}
```

Output:

```
Problems  Javadoc  Declaration  Console
<terminated> stqa02 [Java Application] C:\Users\bipin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe
1.ChromeBrowser
2. Firefox Broswer
Choice
1
Starting ChromeDriver 95.0.4638.17 (a9d0719444d4b035e284ed1fce73bf6ccd789df2-refs/branch-heads/4638@{#178}) on port 9160
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1638902728.464][WARNING]: This version of ChromeDriver has not been tested with Chrome version 96.
Dec 08, 2021 12:15:29 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
```



Conclusion:

Web Drivers on Chrome & Firefox Browsers are implemented.

Practical 3

Aim: Demonstrate handling multiple frames in selenium.

Theory:

iFrame

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 above diagram)
- 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 is meaning to say the page consisting an iframe.

We can even identify total number of iframes by using below snippet.

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

How to switch over the elements in iframes using Web Driver commands:

Basically, we can switch over the elements and handle frames in Selenium using 3 ways.

- **By Index**
- **By Name or Id**
- **By Web Element**

Switch to the frame by index:

Index is one of the attributes for frame handling in Selenium through which we can switch to it. Index of the iframe starts with '0'. Suppose if there are 100 frames in page, we can switch to frame in Selenium by using index.

- driver.switchTo().frame(0);
- driver.switchTo().frame(1);

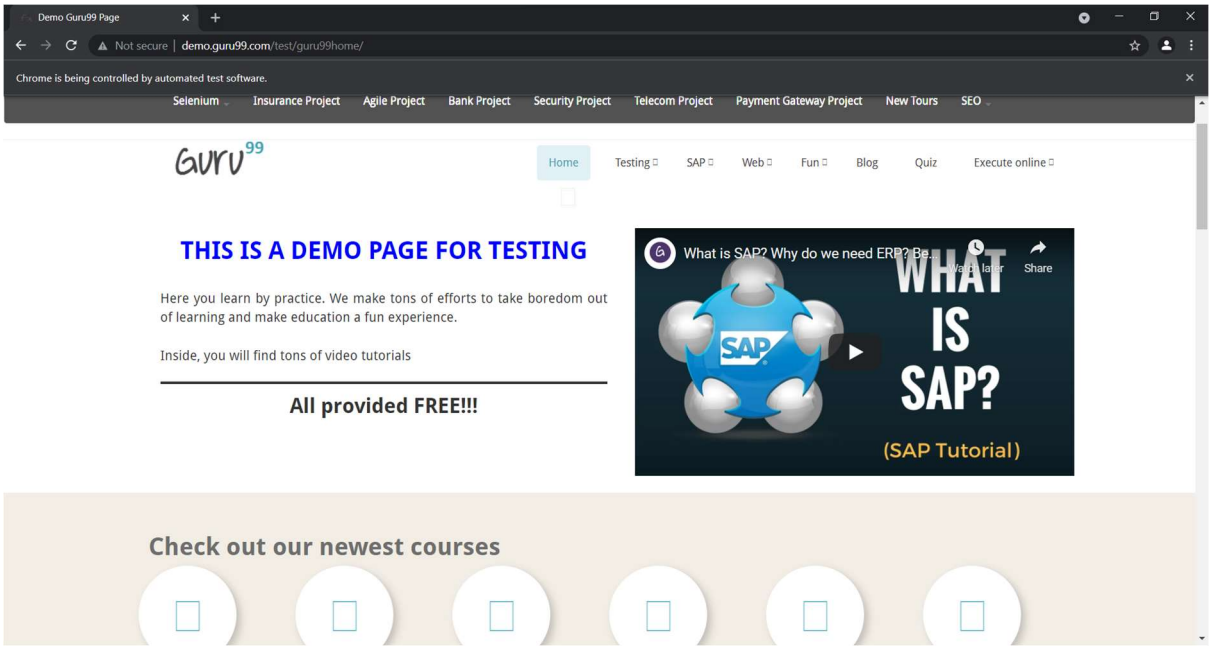
Switch to the frame by Name or ID:

Name and ID are attributes for handling frames in Selenium through which we can switch to the iframe.

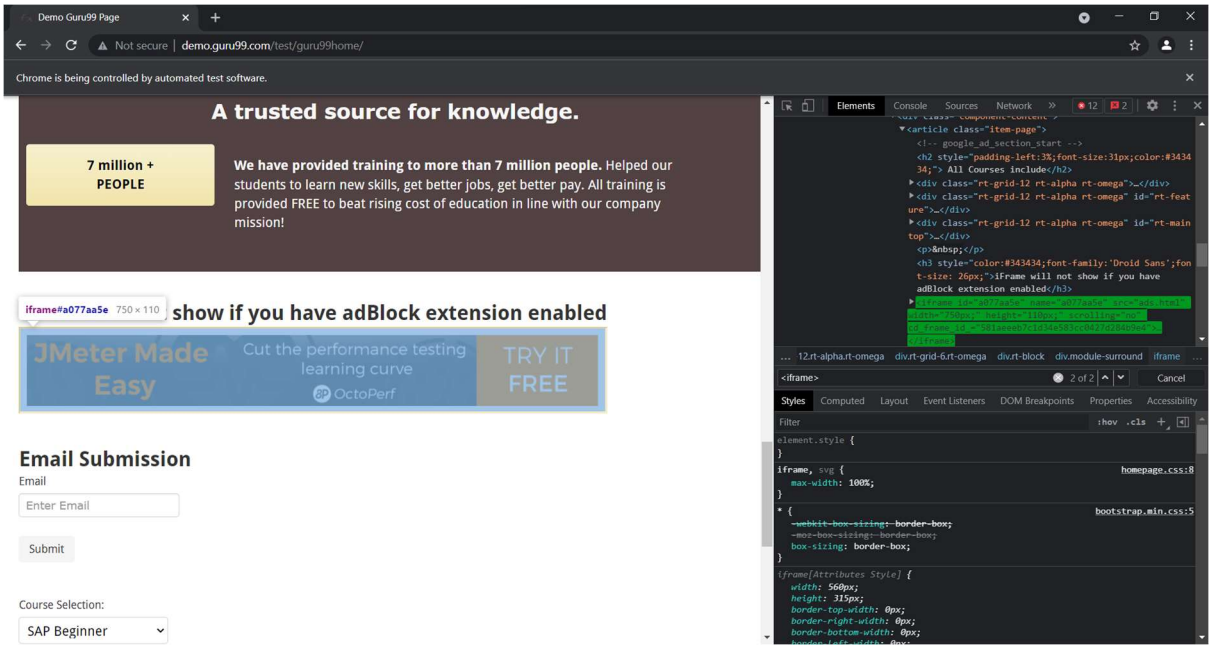
- driver.switchTo().frame("iframe1");
- driver.switchTo().frame("id of the element");

Steps:

1. Go to website



2. Search iframe in inspect



3. Copy frame id from here i.e.(id=" a077aa5e")

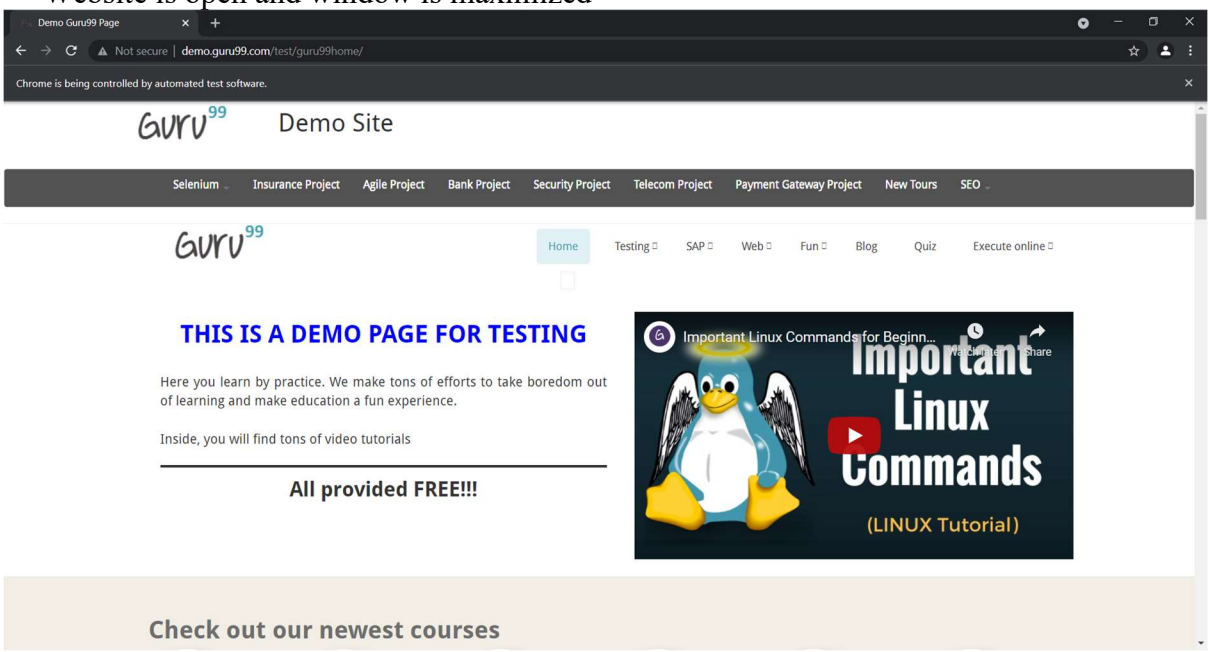
Code:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class stqa03 {
	static WebDriver driver;
	public static void main(String[] args) throws InterruptedException {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\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");
	}
}
```

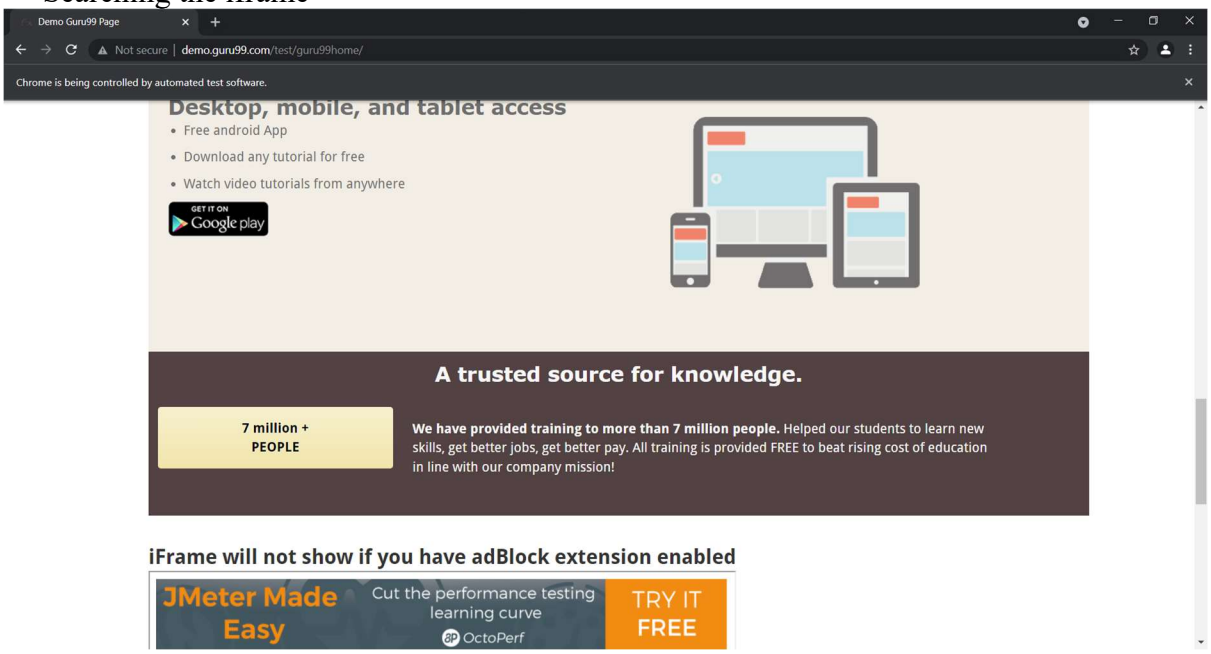
Output:

```
Problems  Javadoc  Declaration  Console
<terminated> stqa03 [Java Application] C:\Users\bipin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (
Starting ChromeDriver 95.0.4638.17 (a9d0719444d4b035e284ed1fce73bf6ccd789df2-refs/branch-heads/4638@{#178}) on port 39771
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1638903725.426][WARNING]: This version of ChromeDriver has not been tested with Chrome version 96.
Dec 08, 2021 12:32:06 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
Switching to iframe
Finished
```

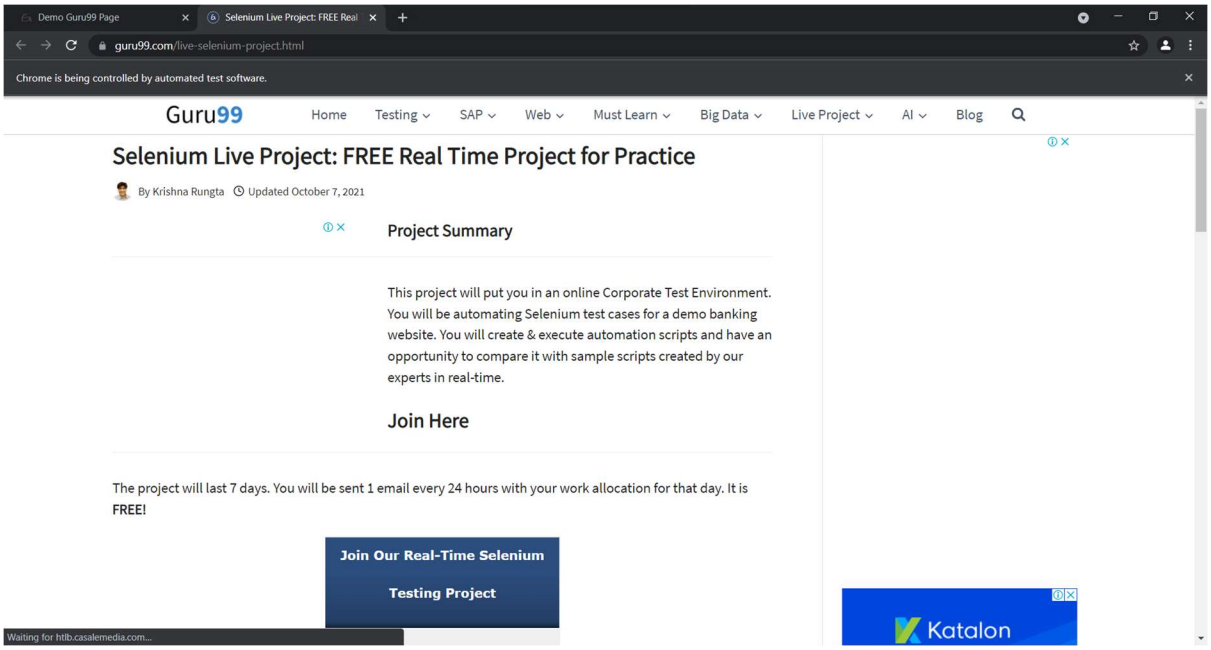
- Website is open and window is maximized



- Searching the iframe



- Clicked on iframe



Conclusion:

Demonstration of handling multiple frames is implemented successfully using 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*.

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

```
driver.get(URL);
// Or can be written as
String URL = "URL";
driver.get(URL);
```

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.

The respective command to fetch the title of the current page can be written as:

```
driver.getTitle();
    // Or can be written as
 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.

The respective command to fetch the string representing the current URL can be written as:

```
driver.getCurrentUrl();
    //Or can be written as
    String CurrentUrl = driver.getCurrentUrl();
```

4. Get Page Source Command
Method:

```
getPageSource(): String
```

In WebDriver, this method returns the source code of the current web page loaded on the current browser. It accepts nothing as parameter and returns a *String* value.
The respective command to get the source code of the current web page can be written as:

```
driver.getPageSource();

//Or can be written as

String PageSource = driver.getPageSource();
```

5. 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*.

The respective command to terminate the browser window can be written as:

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

6. Quit Command
Method:

```
quit(): void
```

This method terminates all windows operating by WebDriver. It terminates all tabs as well as the browser itself. It accepts nothing as parameter and returns void.

The respective command to terminate all windows can be written as:

```
driver.quit();
```

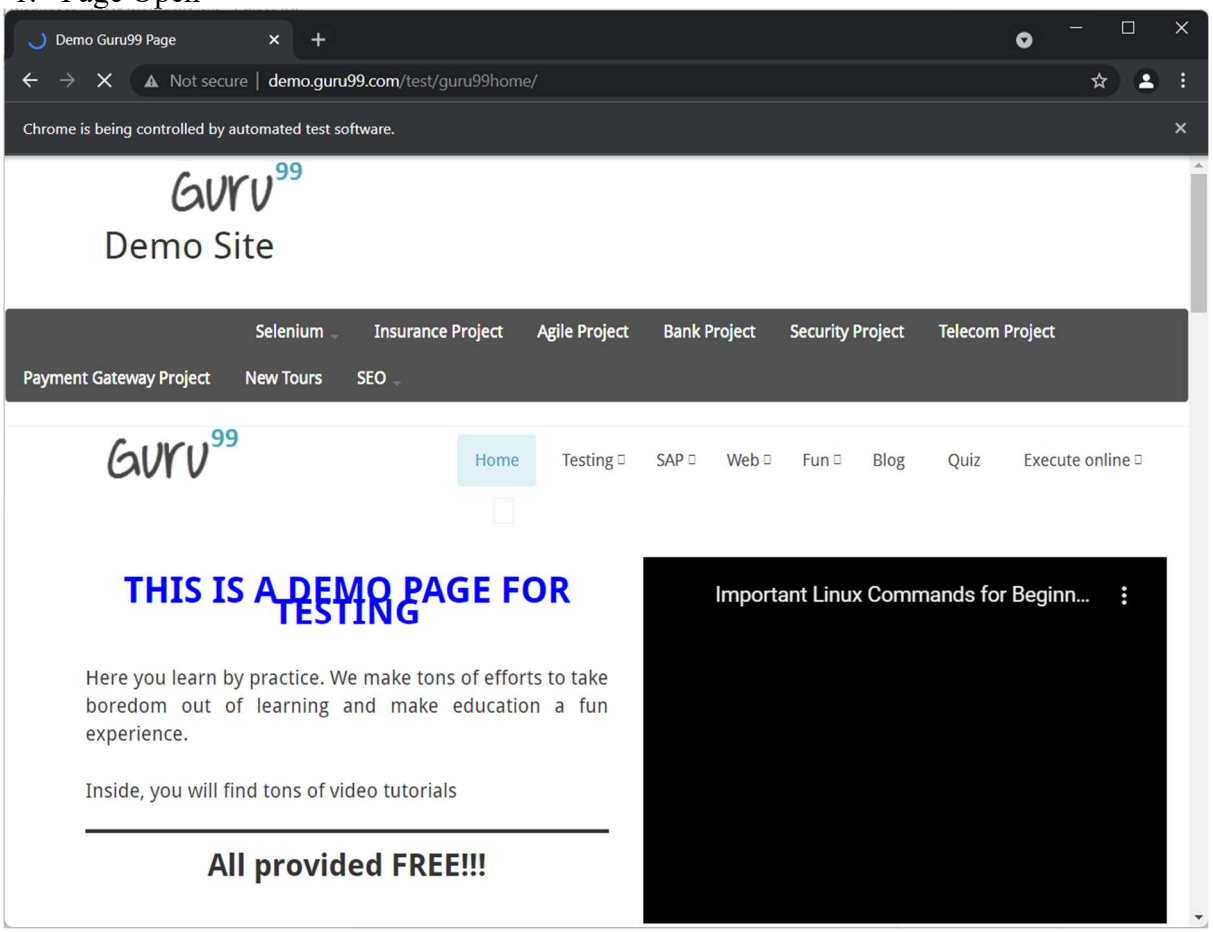
Code:

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

public class stqa04 {
	static WebDriver driver;
	public static void main(String[] args) throws InterruptedException {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		driver = new ChromeDriver();
		String appUrl = "http://demo.guru99.com/test/guru99home/";
		driver.get(appUrl);
		Thread.sleep(2000);
		// Click on Registration link
```

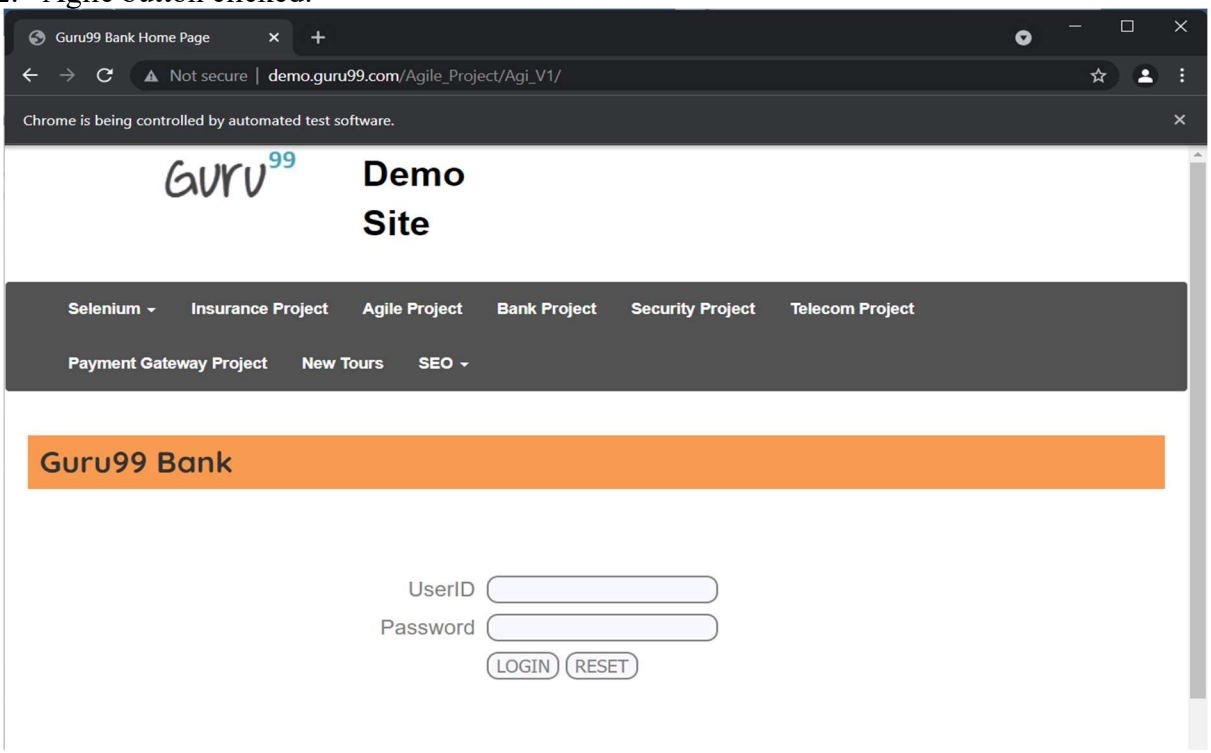
```
        driver.findElement(By.xpath("/html/body/div[1]/div[2]/nav/div/div/ul/li[3]/
a")).click();
                Thread.sleep(2000);
                // Go back to Home Page
                driver.navigate().back();
                Thread.sleep(2000);
                // Go forward to Registration page
                driver.navigate().forward();
                Thread.sleep(2000);
                // Go back to Home page
                driver.navigate().to(appUrl);
                Thread.sleep(2000);
                // Refresh browser
                driver.navigate().refresh();
                Thread.sleep(2000);
                // Close browser
                driver.close();
        }
}
```

Output:

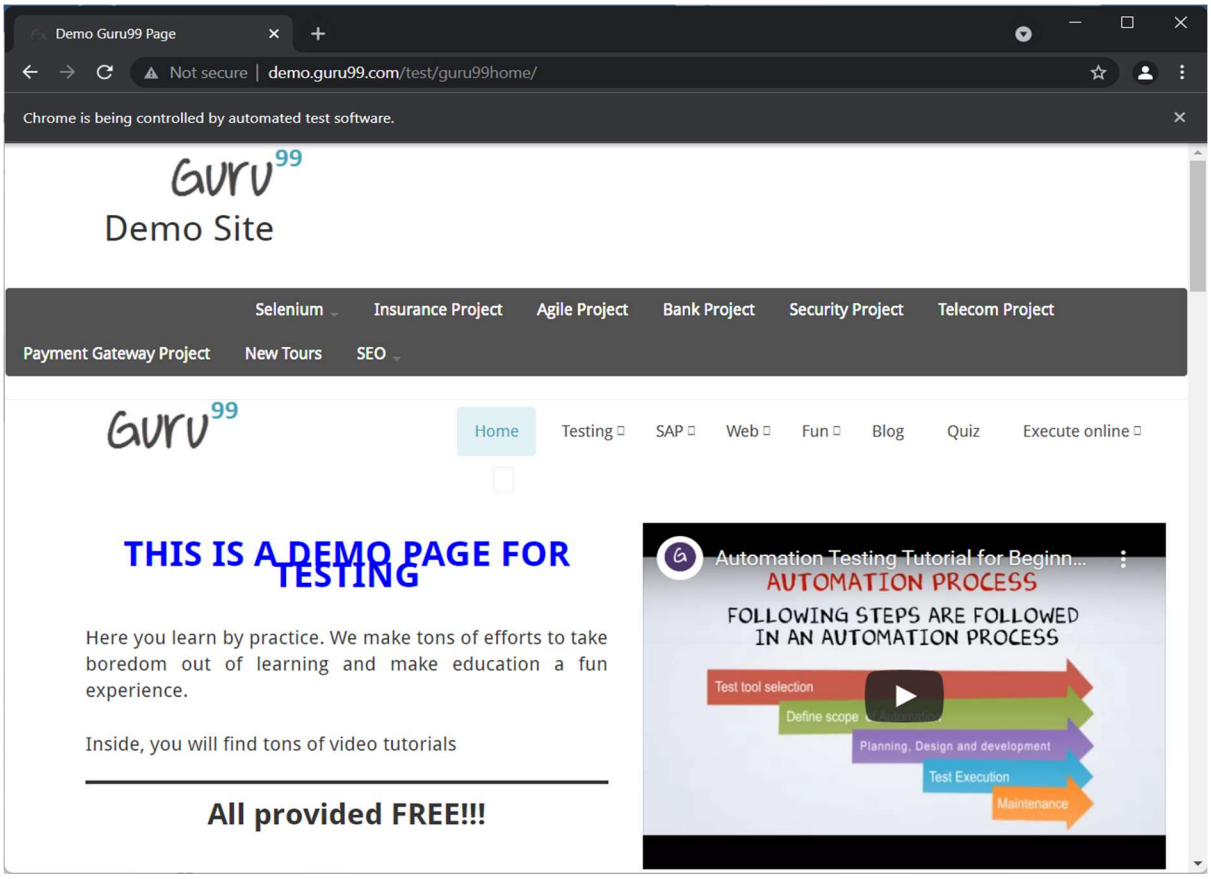
1. Page Open



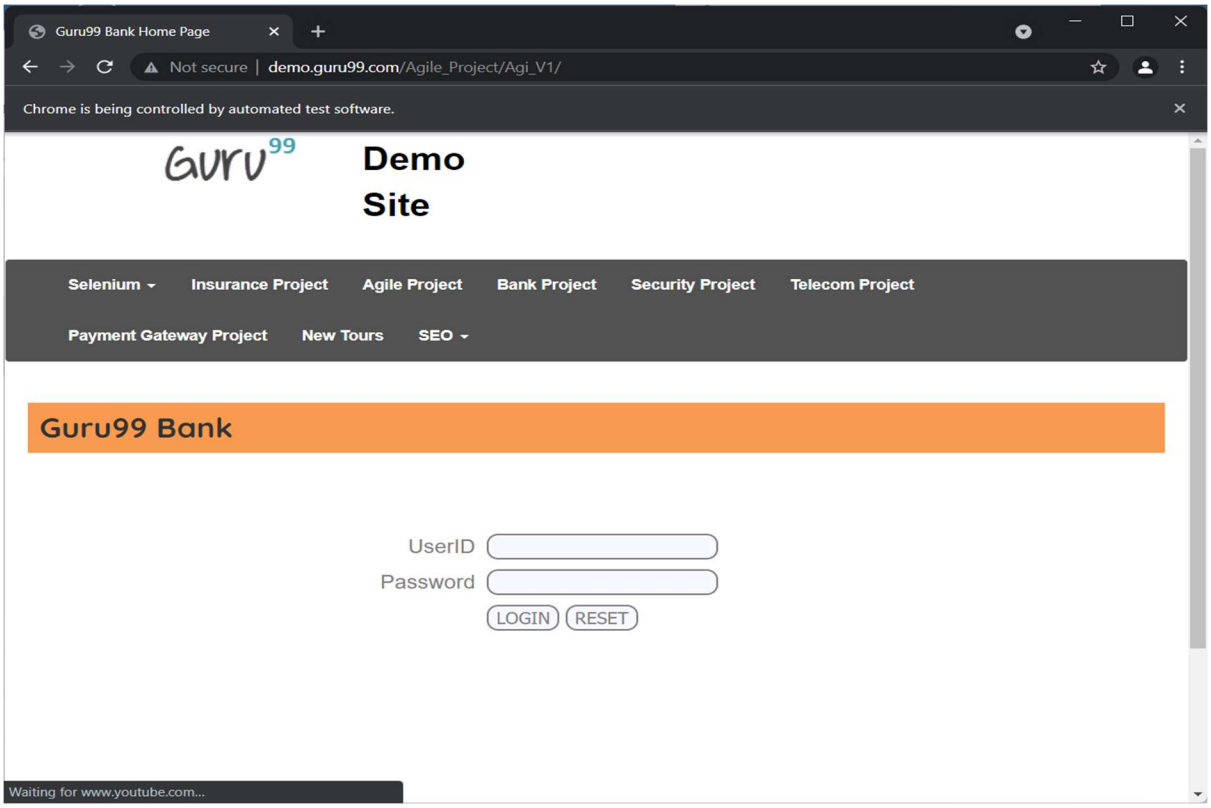
2. Agile button clicked:



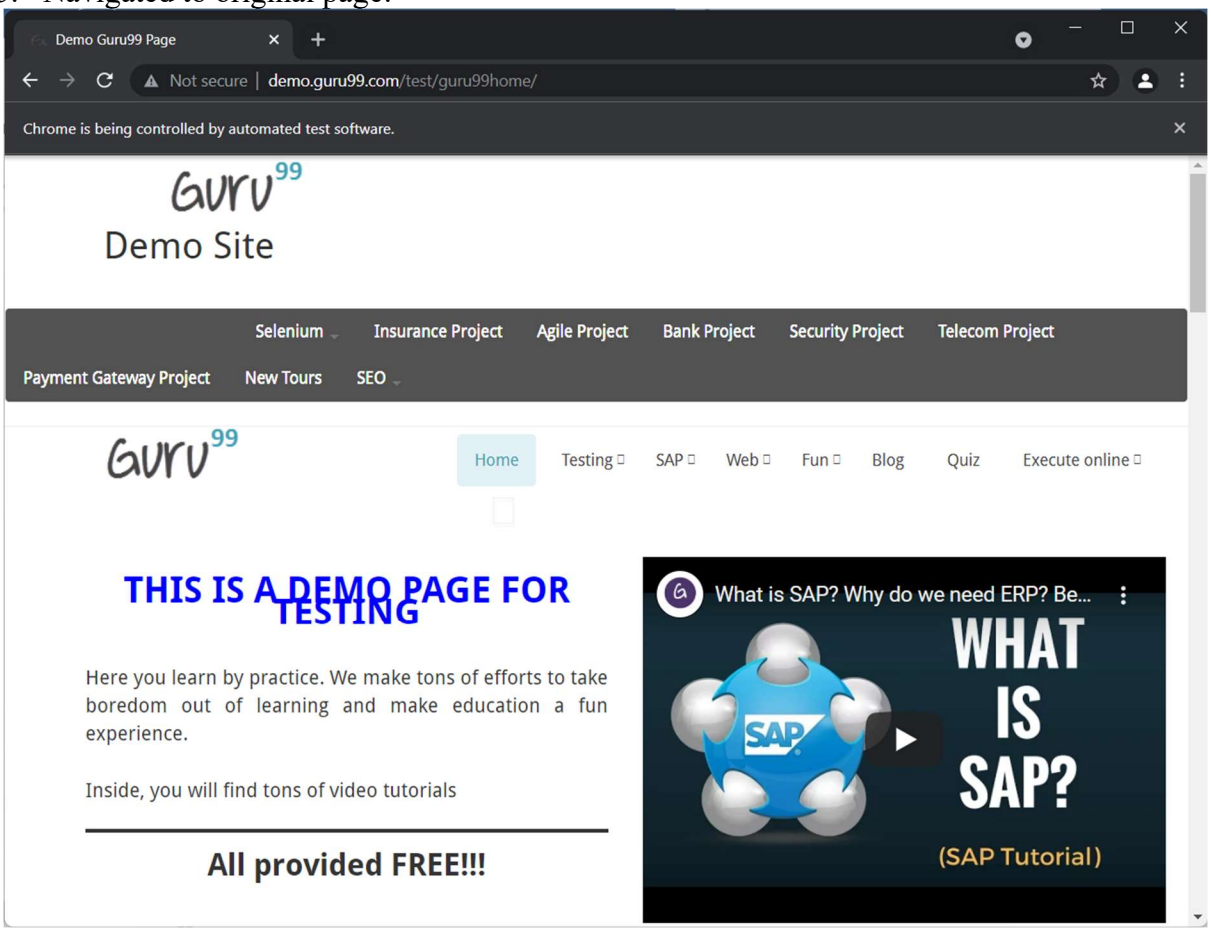
3. Back button clicked:



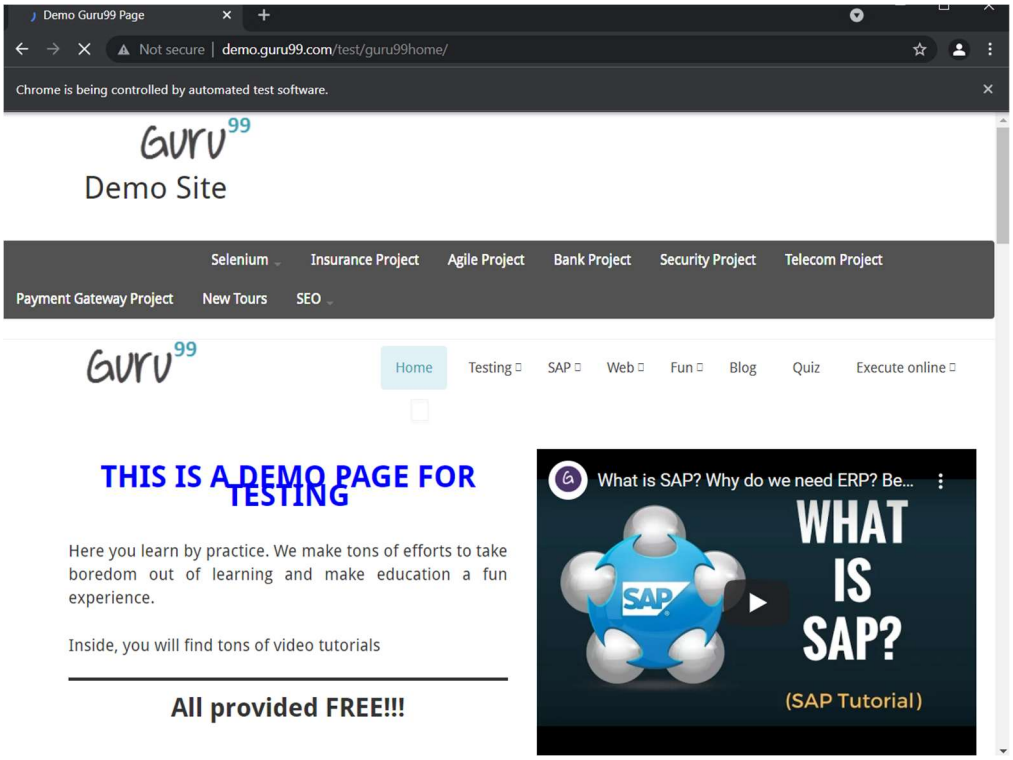
4. Forward button clicked:



5. Navigated to original page:



6. Refresh:



7. Then the browser is closed.

Conclusion:

Browser and navigation commands are implemented.

Practical 5

Aim: Implement the find element command.

Theory:

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 turncan 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.

FindElements:

Find Elements in Selenium command takes in By object as the parameterand 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.

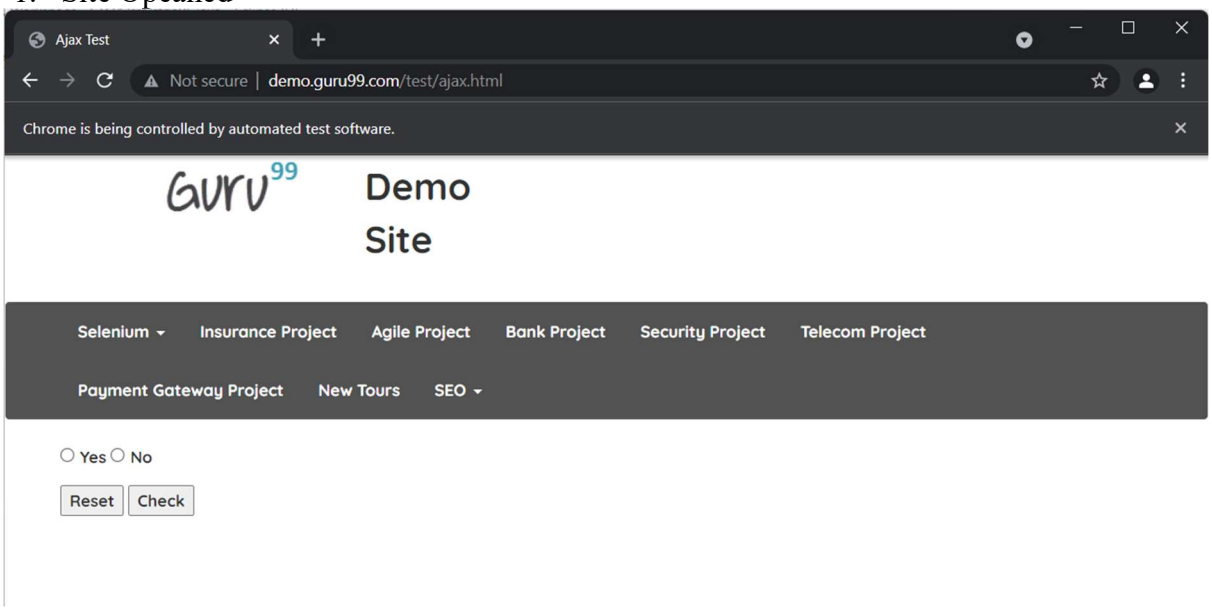
```
List<WebElement> elementName
=driver.findElements(By.LocatorStrategy("LocatorValue"));
```

Code:

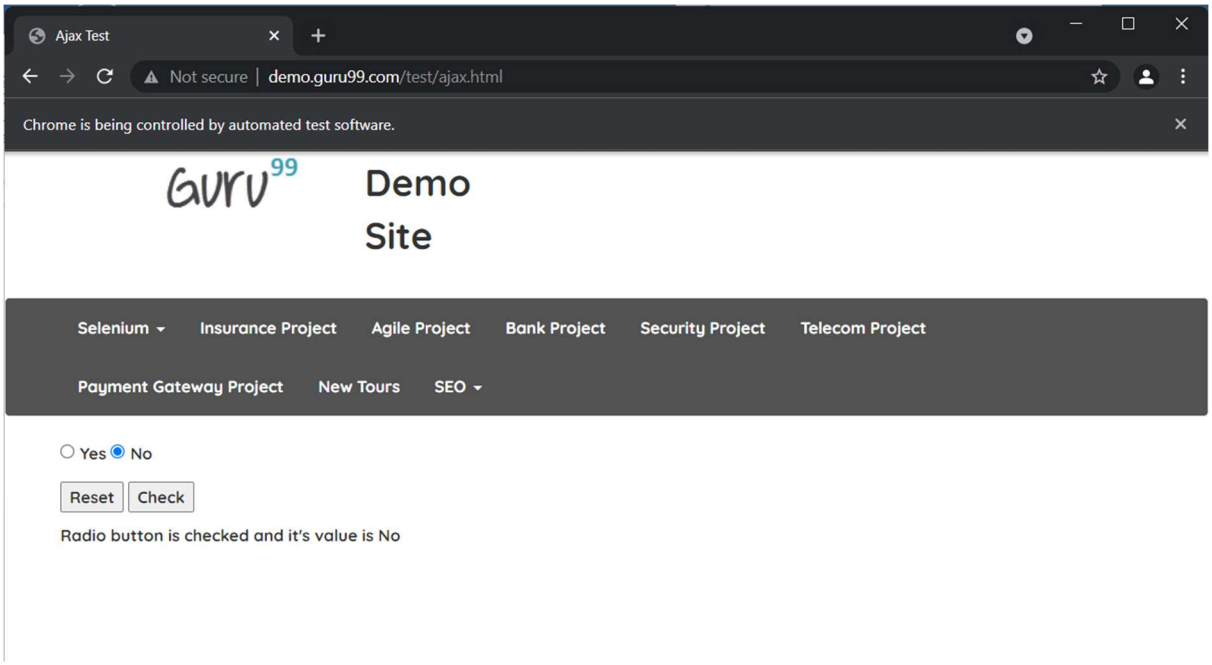
```
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class stqa05 {
	static WebDriver wd;
	public static void main(String[] args) throws InterruptedException {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		wd = new ChromeDriver();
		wd.get("http://demo.guru99.com/test/ajax.html");
		Thread.sleep(2000);
		wd.findElement(By.id("no")).click();
		Thread.sleep(2000);
		wd.findElement(By.id("buttoncheck")).click();
		Thread.sleep(2000);
		List<WebElement> elements = wd.findElements(By.name("name"));
		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:

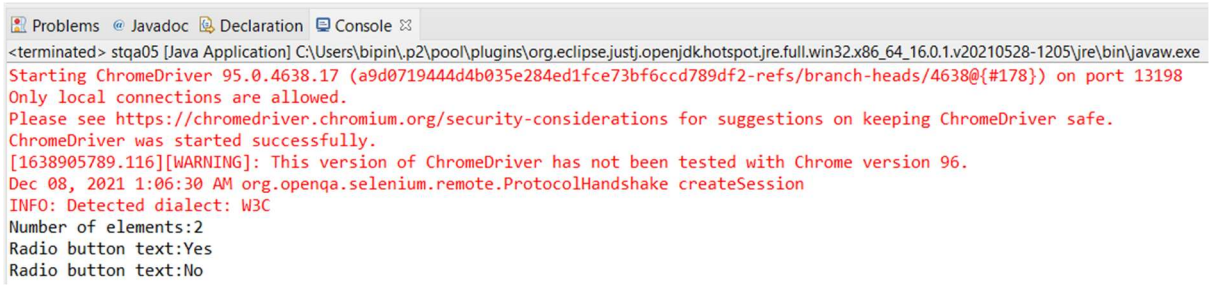
1. Site Opeaned



2. "No" Check box is selected and "Check" Button is clicked



Console:



Conclusion:

Find element command has been successfully implemented.

Practical 6

Aim: Demonstrate the Locator(id, css selector, path).

Theory:

What are Locators?

Locators are the way to identify an *HTML* element on a web page, and almost all UI automation tools provide the capability to use locators for the identification of *HTML* elements on a web page. Following the same trend, Selenium also possesses the ability to use "***Locators***" for the identification of HTML elements and are popularly known as "***Selenium Locators***". Selenium supports various kinds of locators.

Locators are one of the essential components of Selenium infrastructure, which help Selenium scripts in ***uniquely identifying the WebElements***(*such as text box, button, etc.*) present of the web page. So, how do we get the values of these locators? And how to use the same in the automation framework? Let's first understand on the whole page, how we can identify a specific web element in ***DOM*** and then we will try to grab its locator

As we can see, Selenium supports the following locators:

- ***ClassName*** *– A ClassName operator uses a class attribute to identify an object.*
- ***cssSelector*** *– CSS is used to create style rules for webpages and can be used to identify any web element.*
- ***Id*** *– Similar to class, we can also identify elements by using the 'id' attribute.*
- ***linkText*** *– Text used in hyperlinks can also locate element*
- ***name*** *– Name attribute can also identify an element*
- ***partialLinkText*** *– Part of the text in the link can also identify an element*
- ***tagName*** *– We can also use a tag to locate elements*
- ***xpath*** *– Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.*

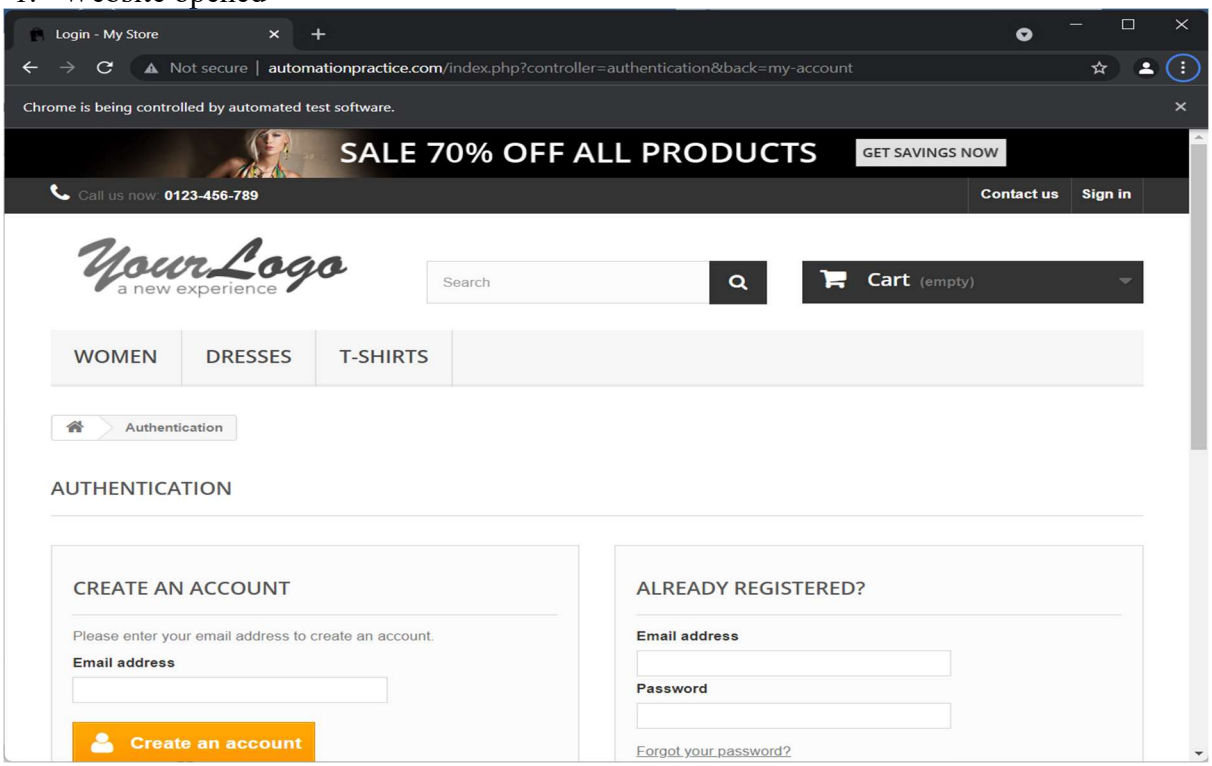
Code:

```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class stqa06 {
	static WebDriver wd;
	public static void main(String[] args) throws InterruptedException {
		int delay=10000;

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		wd = new ChromeDriver();

	wd.get("http://automationpractice.com/index.php?controller=authentication&b
ack=my-account");
		wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);

	wd.findElement(By.name("email")).sendKeys("ronak@gmail.com");//locator id
		Thread.sleep(delay);
```

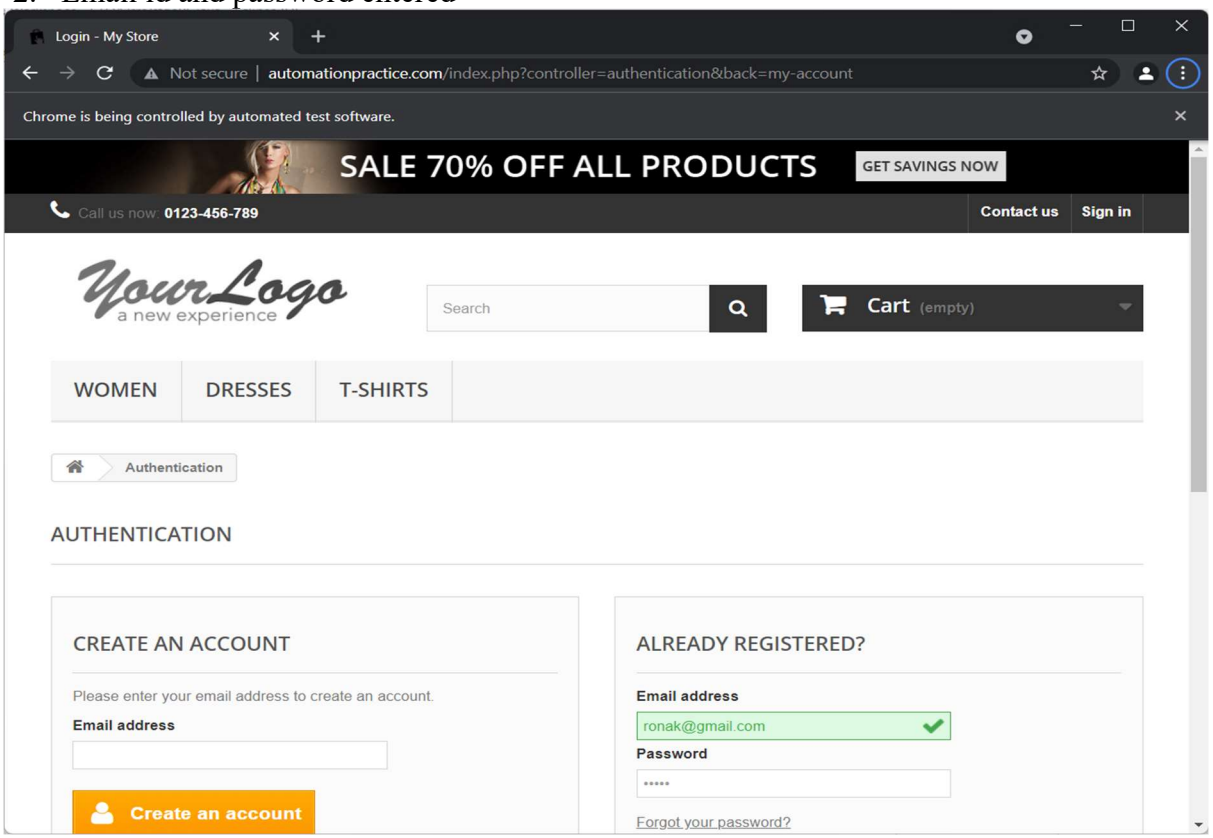
```
            wd.findElement(By.id("passwd")).sendKeys("ronak");//locater name
            Thread.sleep(delay);
            wd.findElement(By.id("SubmitLogin")).click();//locator className
            Thread.sleep(delay);
            //CSS selector
            //tagname#id

      wd.findElement(By.cssSelector("input#search_query_top")).sendKeys("Flowers"
);
            Thread.sleep(delay);
            //tagname[attribute=value of attribute]
            wd.findElement(By.cssSelector("button[type=submit]")).click();
            Thread.sleep(delay);
            //tabname.classname
            wd.findElement(By.cssSelector("a.logout")).click();
            Thread.sleep(delay);
            //XPATH
            //tagname[@attribute=value of attribute]

      wd.findElement(By.xpath("/html/body/div/div[1]/header/div[2]/div/div/nav/di
v[1]/a")).click();
            //relative xpath by self writing
      }
}
```

Output:

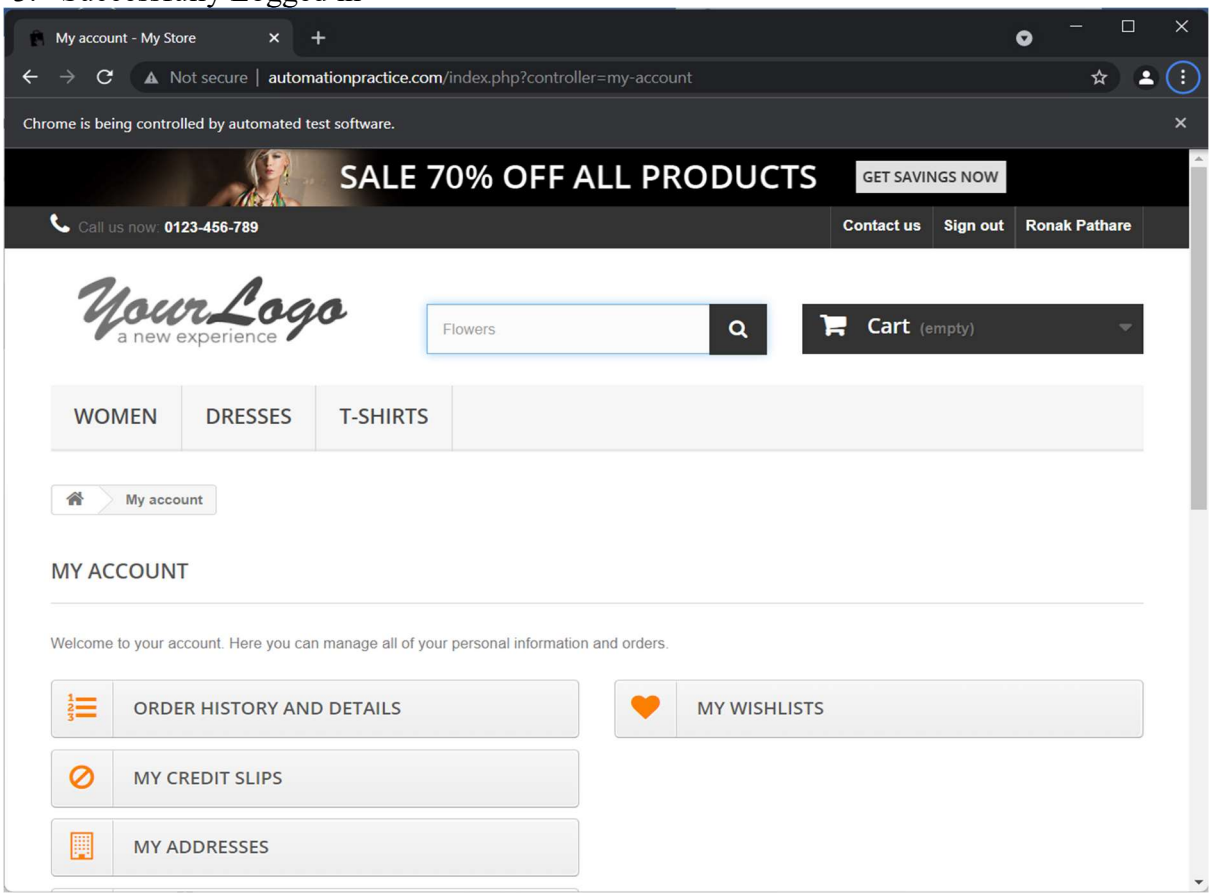
1. Website opened



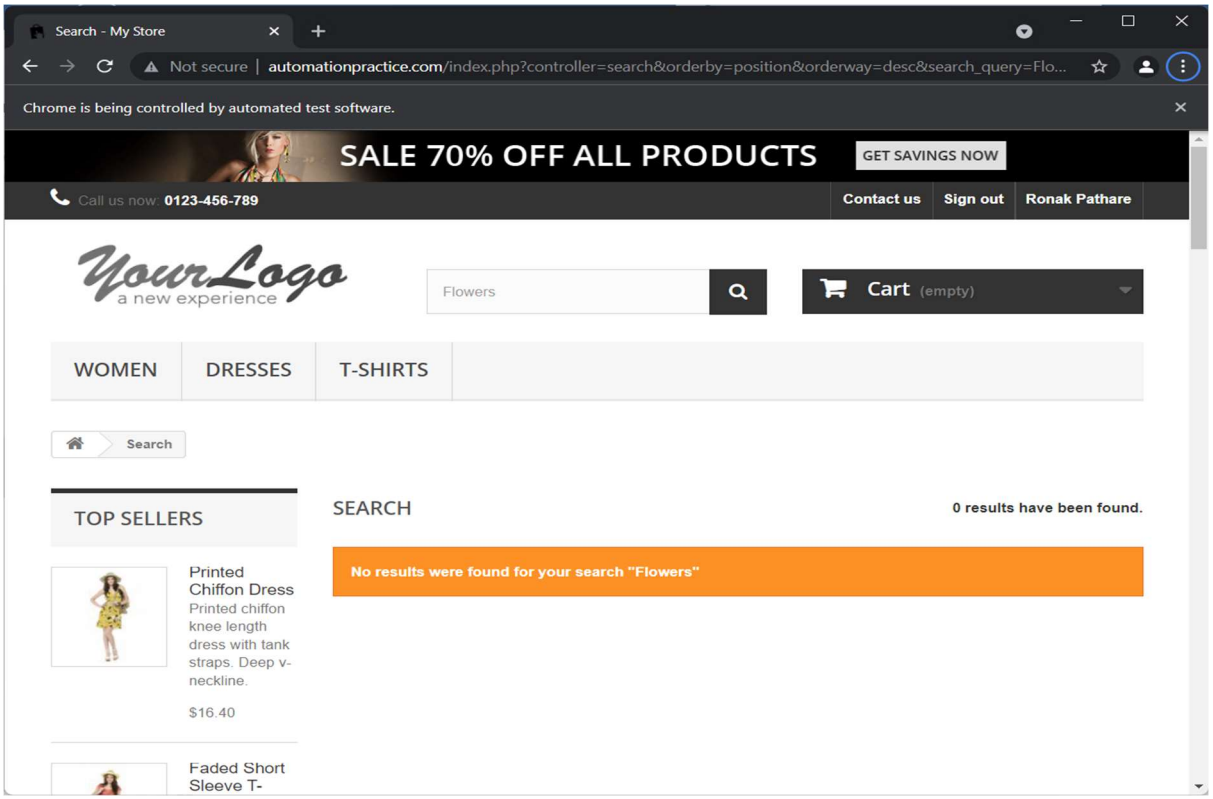
2. Email id and password entered



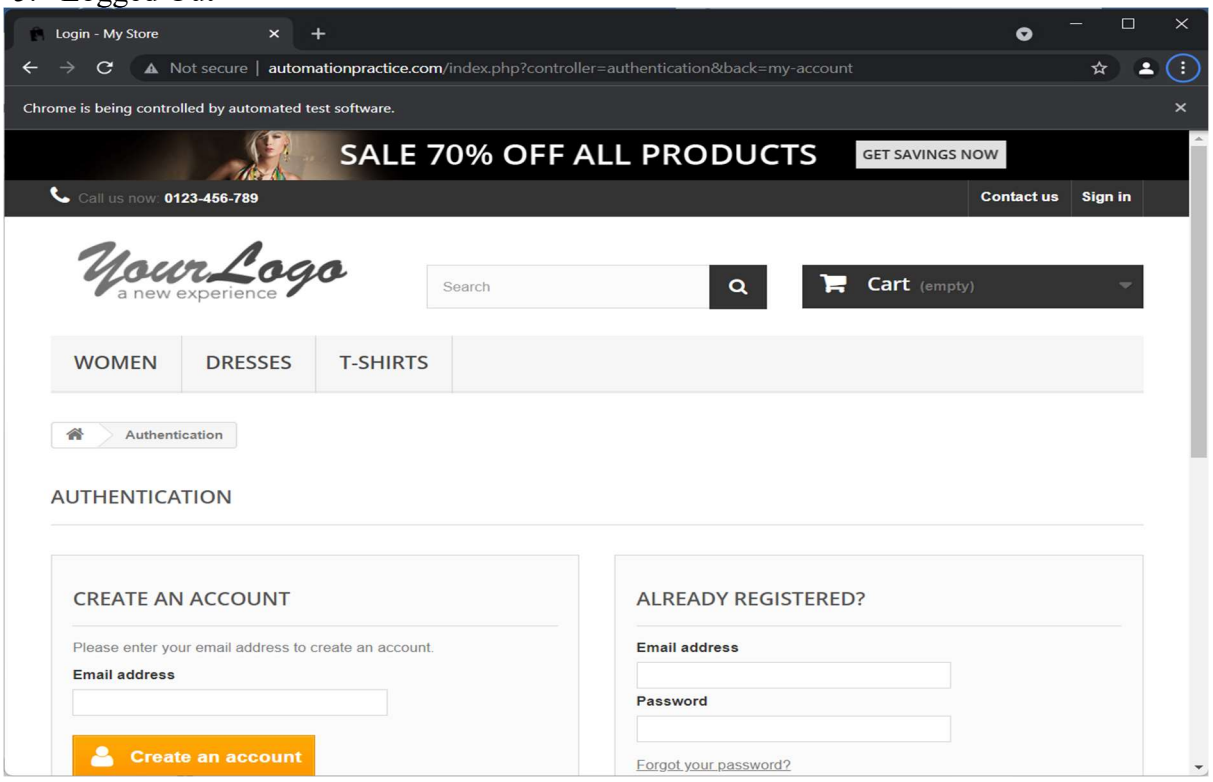
3. Successfully Logged in



4. Searched foe “Flowers”



5. Logged Out



Conclusion:

Locator based on id, css selector and xpath has been demonstrated.

Practical 7

Aim: Demonstrate synchronization in selenium.

Theory:

Synchronization in **selenium:**

Synchronization meaning when two or more components involved to perform any action, we expect these components to work together with the same pace. The co-ordination between these components to run parallelly is called Synchronization.

Synchronization (Wait) in Selenium has a great significant value.

Now a day, we see most of the web applications are developed by using Javascript and Ajax where it might happen that some of the elements may load at distant time intervals, due to which we may encounter "ElementNotVisibleException" or "NoSuchElementException" exceptions. In such scenarios, we use Synchronization/wait to overcome these kinds of exceptions.

Automation Testing is commonly comprising of following two segments:

1. **Application Under Test**
2. **Test Automation Tool.**

Synchronization can be classified into two categories:

1. **Unconditional Synchronization**

   Unconditional synchronization indicates timeout value alone and makes tool to continue further only when specified time is elapsed. One such functions is:

   System.Threading.Thread.Sleep();

   This statement is helpful only when we are interacting with third party systems like interfaces because it is not possible to write conditions to check the status. Now, we want tool to wait for certain amount of time and then perform the next actions sequence.

   This statement makes the tool to wait for certain time even though the application is ready, thus increasing the chance of unnecessary wait and hence giving the scope to **Conditional Synchronization.**

2. **Conditional Synchronization**

   Conditional Synchronization: By using conditional synchronization we can specify timeout duration along with some desired conditions to check and then throw an error if nothing happens.

Types of Conditional Synchronization:

1. **Implicit Wait**

   Implicit Wait tells the WebDriver to Wait until the stated time before throwing the NoSuchElement/ElementNotVisible exception. Waiting time across the test script between each consecutive steps are taken by default. Hence, next testStep will execute only when the specified time is elapsed post executing the previous testStep.

   Syntax:

driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(30));

The Implicit wait takes one argument: TimeSpan (timeToWait)

- FromMilliSeconds
- FromSeconds
- FromMinutes
- FromHours
- FromDays

2. **Explicit Wait**

Explicit waits are very good to use when page loads dynamically. Explicit Wait tells the WebDriver to Wait until the specified condition is met or maximum time elapses before throwing NoSuchElement (or) ElementNotVisible Exceptions. Explicit waits are applied for the specified testStep in test script.

To use Explicit waits, we must first create instance for **"WebDriverWait"** class.

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(30));
wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.Id(element_ID)));

The reference variable "wait" informs the WebDriver to wait until the Expected condition to occur (or) Wait for the specified time of 30seconds, whichever shows in first place before throwing an exception.

Following are the **ExpectedConditions:**

- AlertIsPresent()
- ElementSelectionStateToBe()
- VisibilityOfAllElementsLocatedBy()
- ElementToBeSelected()
- FrameToBeAvaliableAndSwitchToIt()
- InvisibilityOfTheElementLocated()
- InvisibilityOfElementWithText()
- ElementToBeClickable()

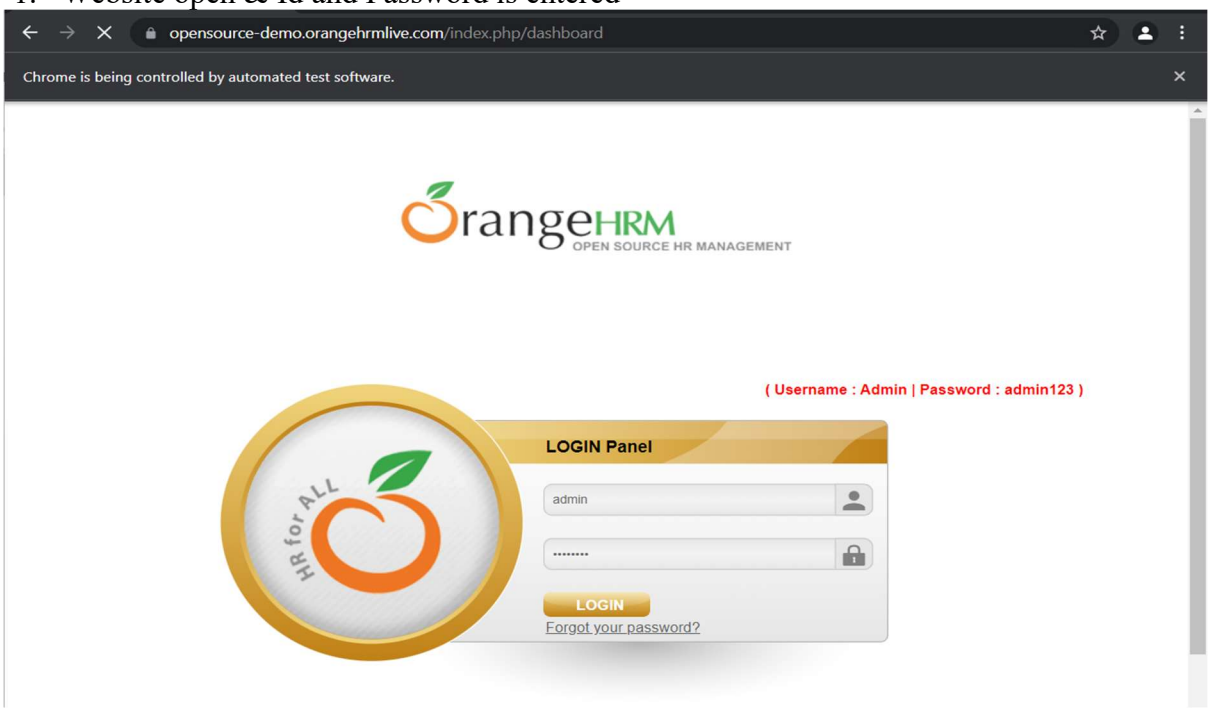
Code:

Implicit Wait:

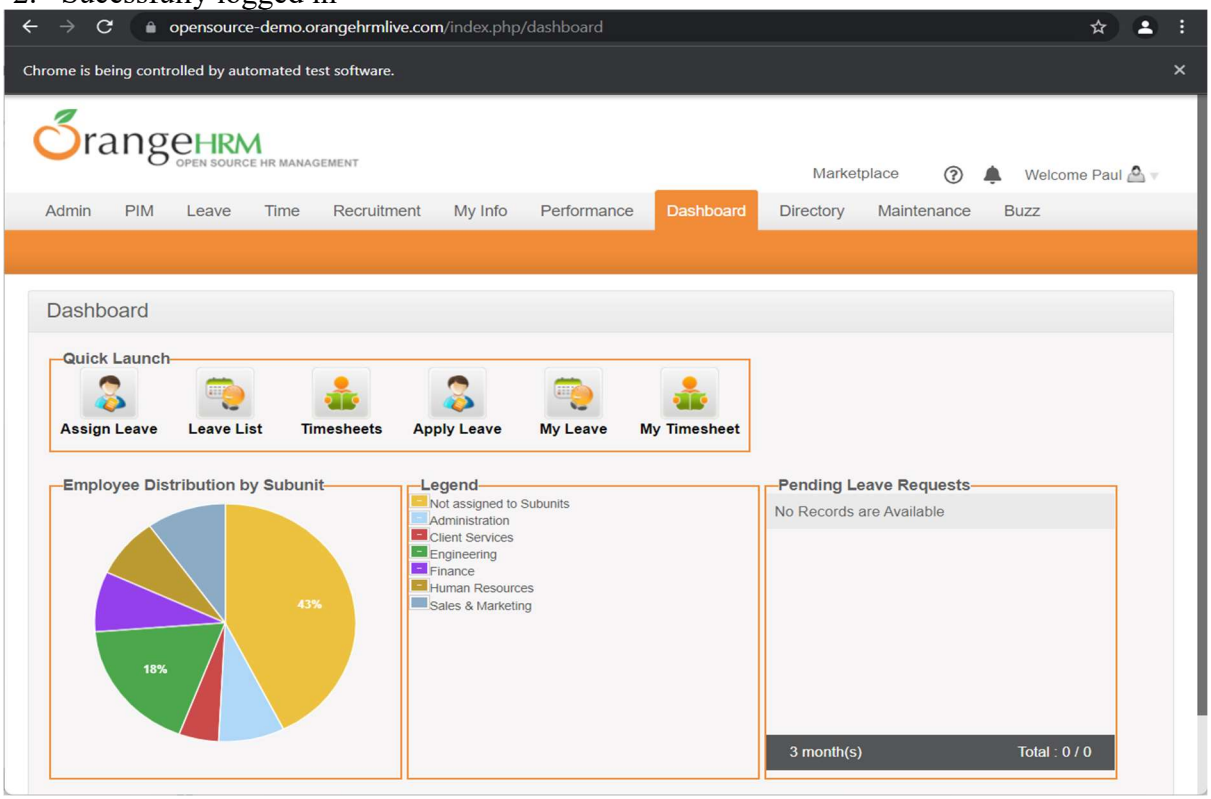
```
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class stqa07A {
	static WebDriver wd;
	public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		wd = new ChromeDriver();
		wd.get("https://opensource-demo.orangehrmlive.com/");
		wd.findElement(By.id("txtUsername")).sendKeys("admin");//locator id
	wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);//implicit wait

	wd.findElement(By.name("txtPassword")).sendKeys("admin123");//locater name
		wd.findElement(By.className("button")).click();//locator className
		wd.findElement(By.partialLinkText("Welcome")).click();//locator
partiallinkText
		wd.findElement(By.xpath("//*[@id=\"welcome\"]")).click();
		wd.findElement(By.linkText("Logout")).click();//locator linkText
	}}
```

Output:

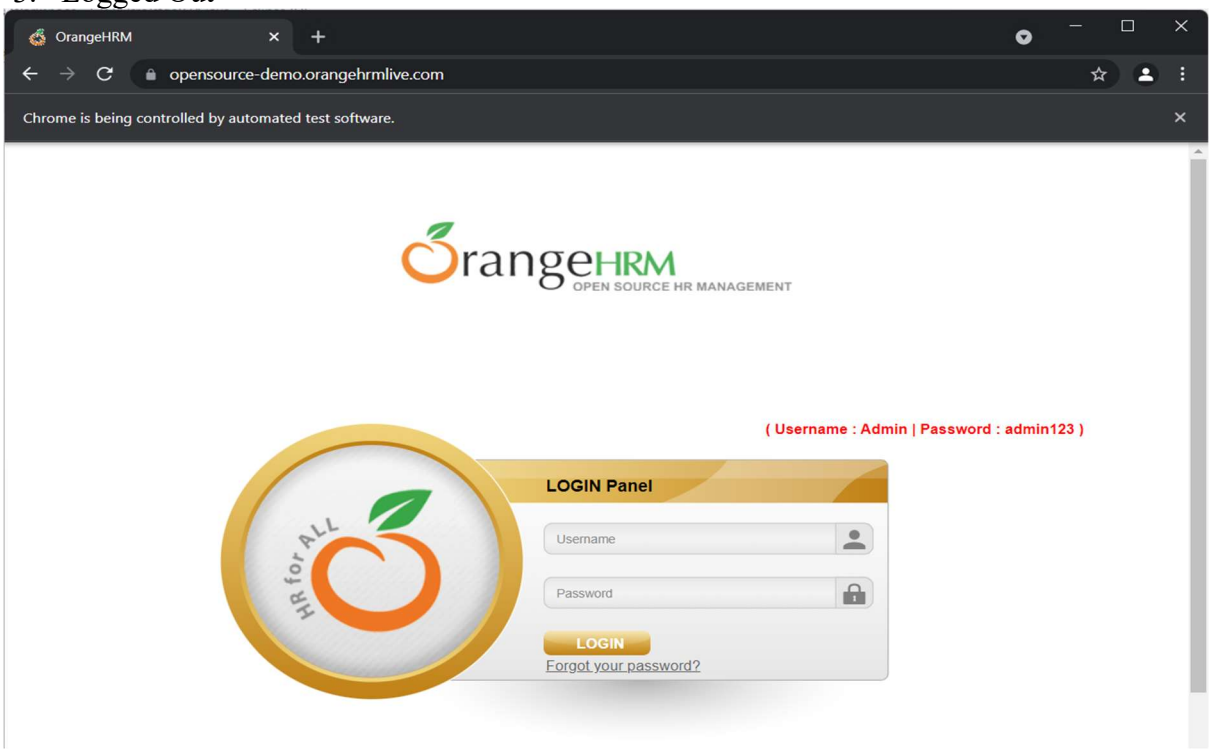
1. Website open & Id and Password is entered



2. Sucessfully logged in



3. Logged Out

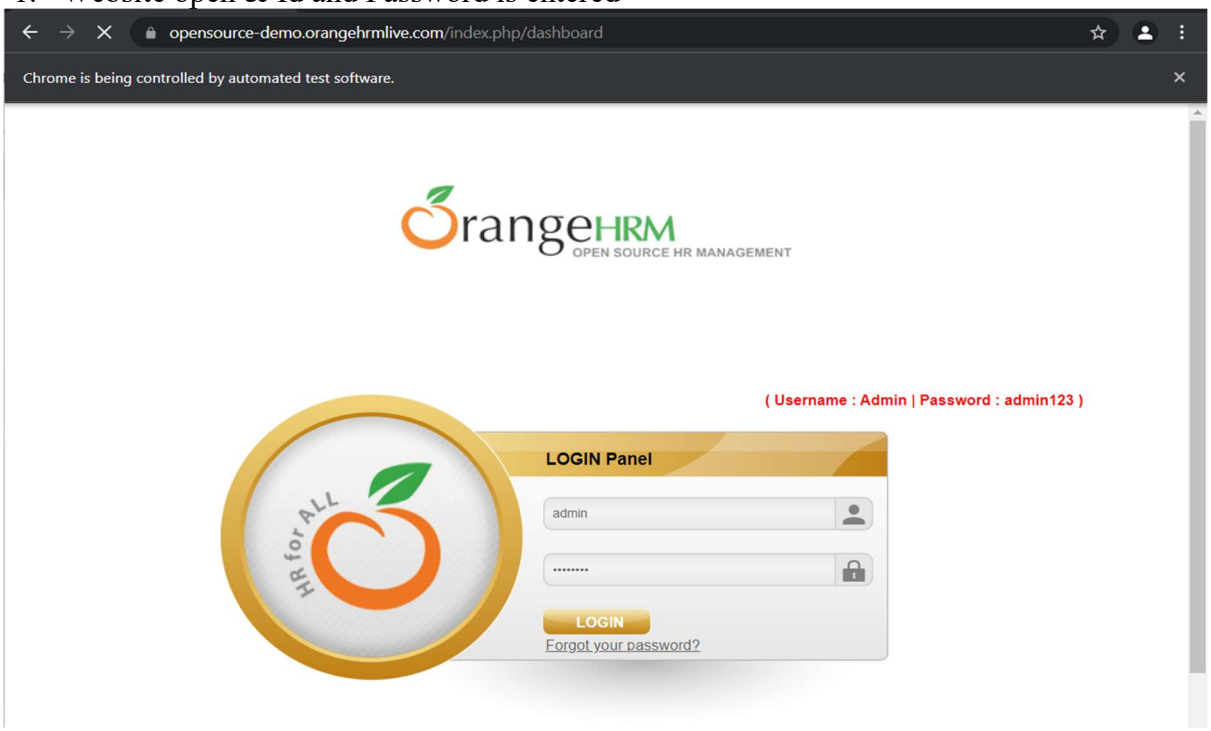


Explicit Wait

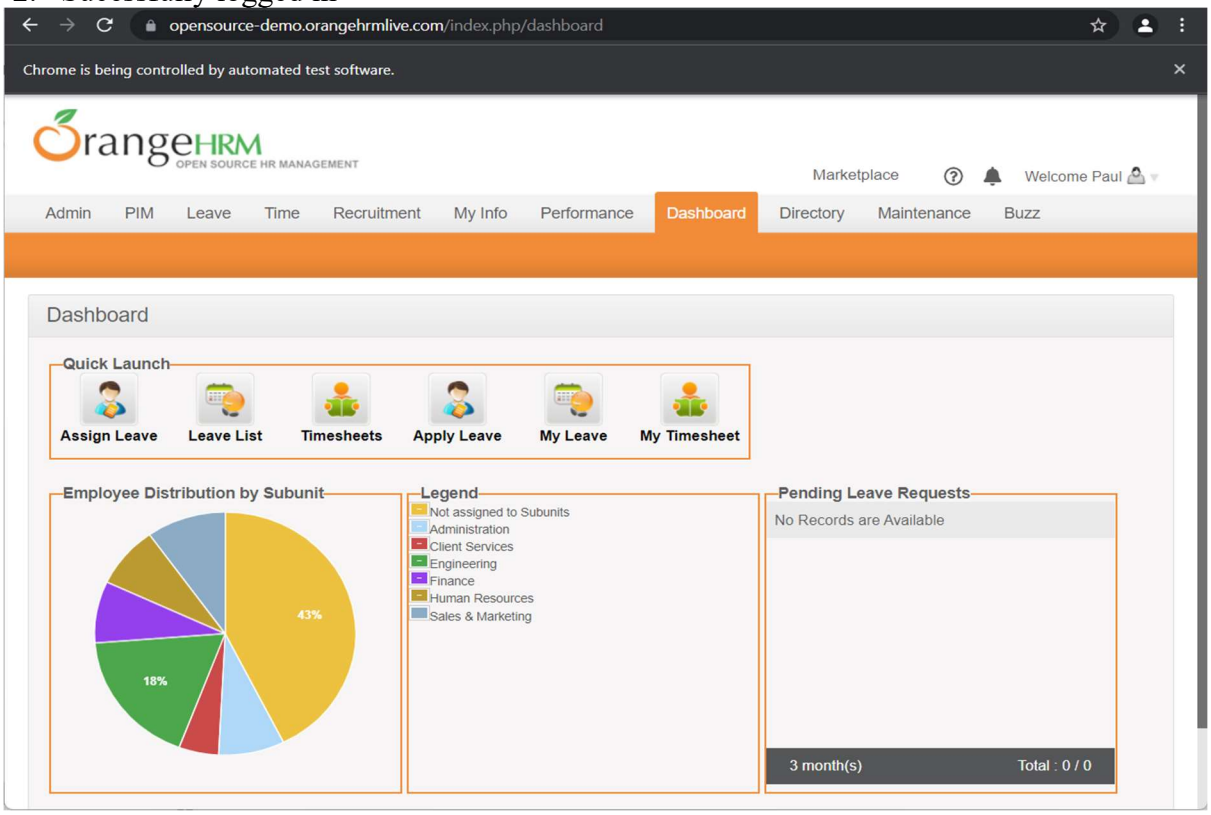
```
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

public class stqa07B {
	public static void main(String[] args) throws InterruptedException {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver wd = new ChromeDriver();
		wd.get("https://opensource-demo.orangehrmlive.com/");
		WebDriverWait wt = new WebDriverWait(wd,30);
		wd.findElement(By.id("txtUsername")).sendKeys("admin");//locator id

	wd.findElement(By.name("txtPassword")).sendKeys("admin123");//locater name
		wd.findElement(By.className("button")).click();//locator className
		wd.findElement(By.partialLinkText("Welcome")).click();//locator
partiallinkText
		wd.findElement(By.xpath("//*[@id=\"welcome\"]")).click();

	wt.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Logout"
)));
		wd.findElement(By.linkText("Logout")).click();//locator linkText
		}
}
```

Output:

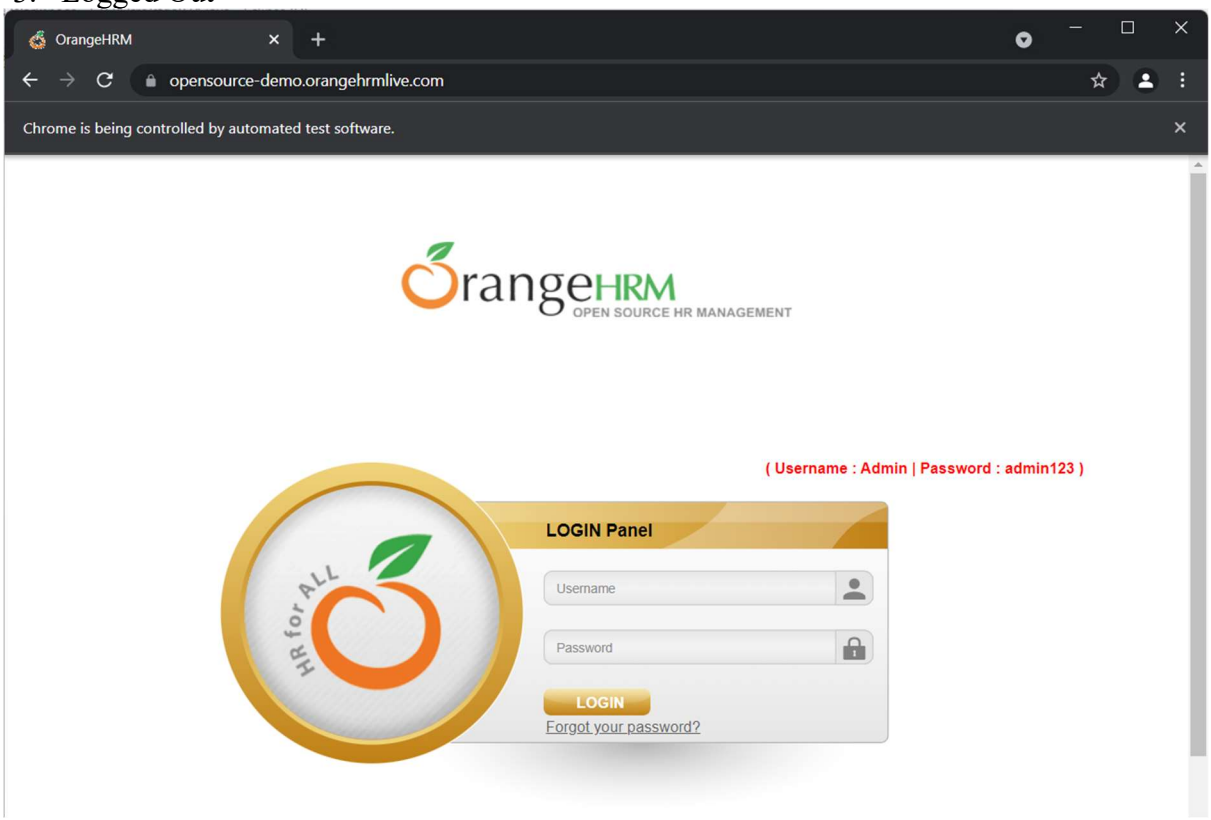
1. Website open & Id and Password is entered



2. Sucessfully logged in



3. Logged Out



Conclusion:

Implicit wait and explicit wait have been demonstrated

Practical 8

Aim: Demonstrate different types of alerts.

Theory:

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.

Here are few alert in Selenium types:

1) Simple Alert

The simple alert class in Selenium displays some information or warning on the screen.

2) Prompt Alert.

This Prompt Alert asks some input from the user and Selenium webdriver can enter the text using sendkeys(" input…. ").

3) Confirmation Alert.

This confirmation alert asks permission to do some type of operation.

How to handle Alert in Selenium WebDriver

Alert interface provides the below few methods which are widely used in Selenium Webdriver.

1) void dismiss() // **To click on the 'Cancel' button of the alert.**

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

2) void accept() // **To click on the 'OK' button of the alert.**

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

3) String getText**()** // **To capture the alert message.**

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

4) void sendKeys(String stringToSend) // **To send some data to alert box.**

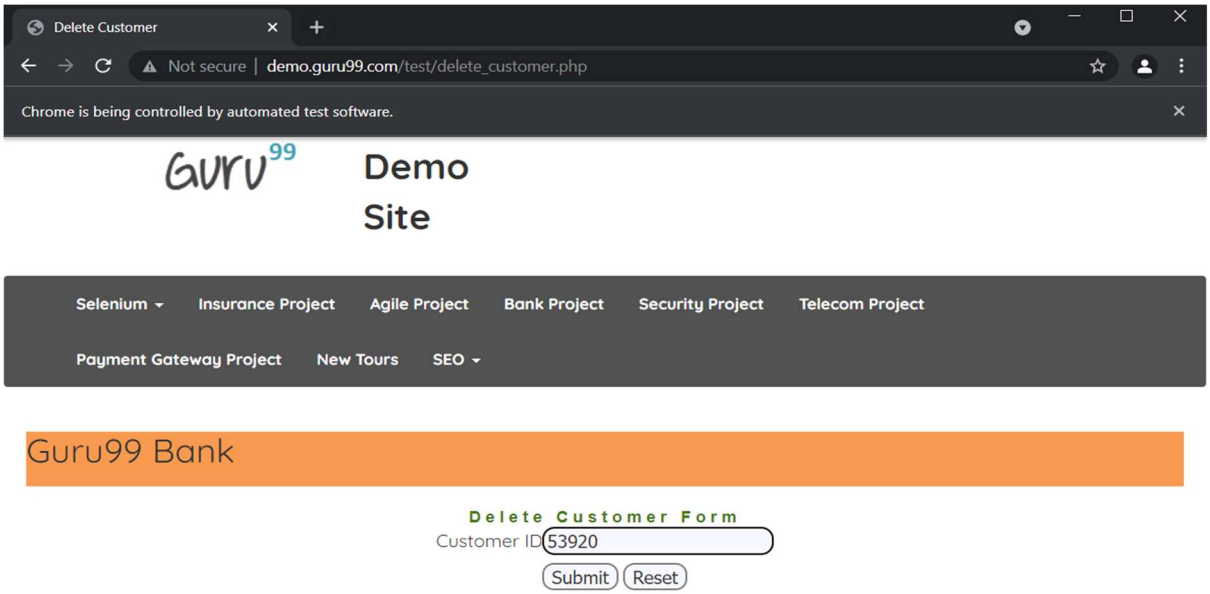
```
driver.switchTo().alert().sendKeys("Text");
```

Code:

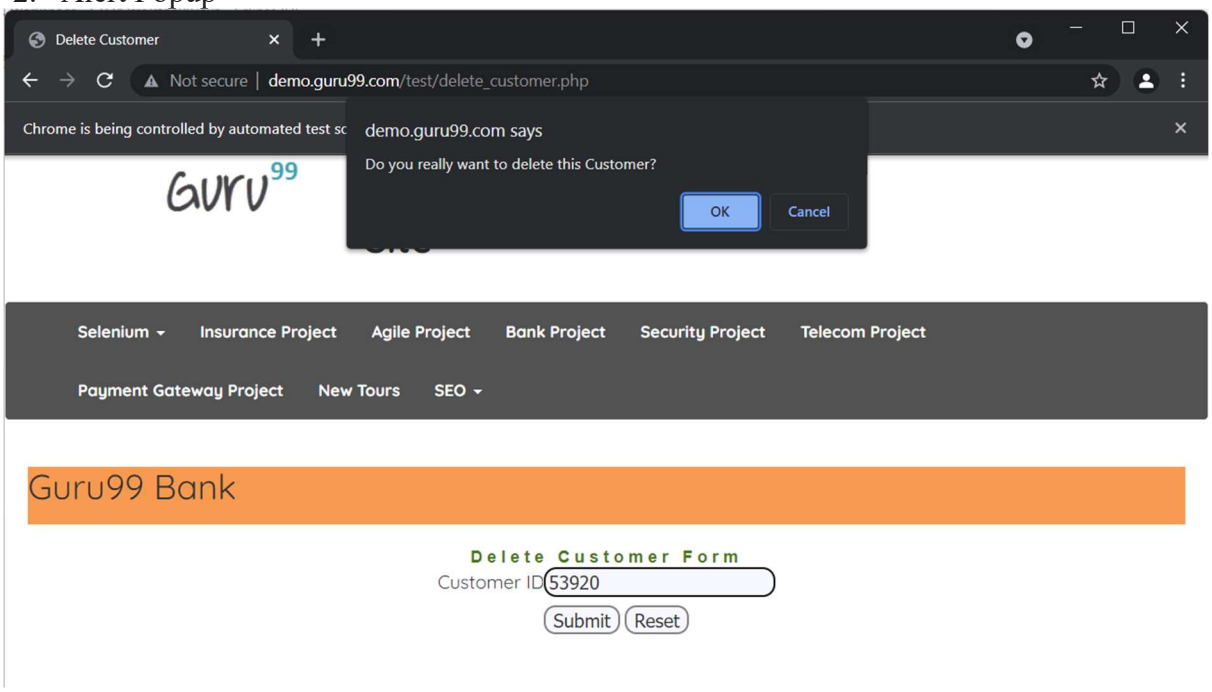
```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.NoAlertPresentException;
import org.openqa.selenium.Alert;

public class stqa08 {
	public static void main(String[] args) throws
	NoAlertPresentException,InterruptedException {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.get("http://demo.guru99.com/test/delete_customer.php");
		driver.findElement(By.name("cusid")).sendKeys("53920");
		driver.findElement(By.name("submit")).click();
		// Switching to Alert
		Alert alert = driver.switchTo().alert();
		// Capturing alert message.
		String alertMessage= driver.switchTo().alert().getText();
		// Displaying alert message
		System.out.println(alertMessage);
		Thread.sleep(5000);
		// Accepting alert
		alert.accept();
	}
}
```

Output:

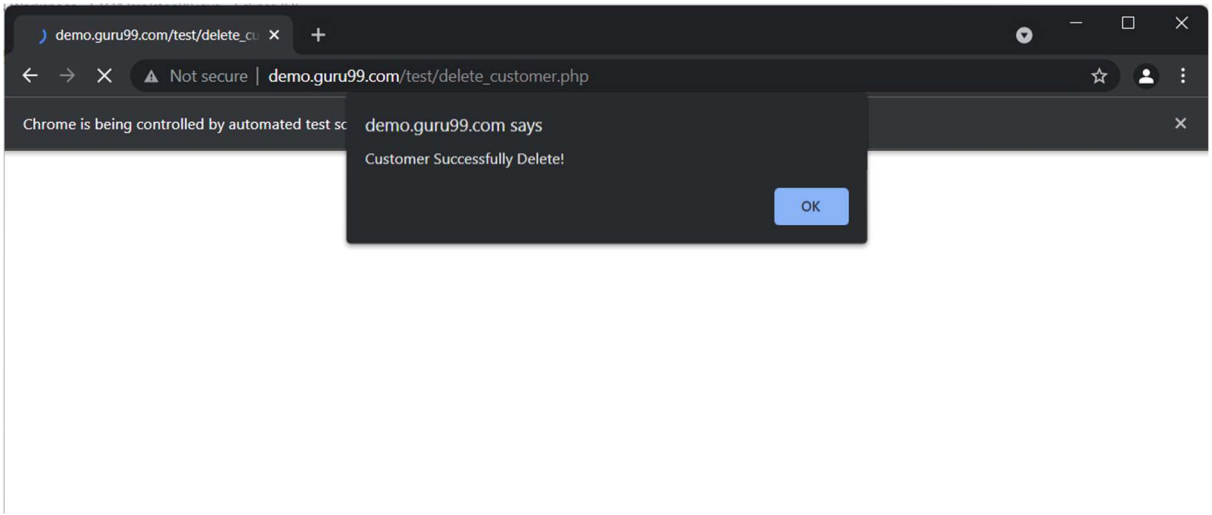
1. Site open and Customer id is entered



2. Alert Popup



3. Customer Deleted



Conclusion:

Concept of alert has been successfully demonstrated.

Practical 9

Aim: Demonstrate Handling Drop Down and List Boxes.

Theory:

Select Class in Selenium:

The 'Select' class in Selenium WebDriver is used for selecting and deselecting option in a dropdown. The objects of Select type can be initialized by passing the dropdown web Element as parameter to its constructor.

```
WebElement testDropDown = driver.findElement(By.id("testingDropdown"));
Select dropdown = new Select(testDropDown);
```

How to select an option from drop-down menu?

WebDriver provides three ways to select an option from the drop-down menu.

1. selectByIndex - It is used to select an option based on its index, beginning with 0.

```
dropdown.selectByIndex(5);
```

2. selectByValue - It is used to select an option based on its 'value' attribute.

```
dropdown.selectByValue("Database");
```

3. selectByVisibleText - It is used to select an option based on the text over the option.

```
dropdown.selectByVisibleText("Database Testing");
```

Let us consider a test case in which we will automate the following scenarios:

- Invoke Google Chrome Browser
- Open URL: https://www.testandquiz.com/selenium/testing.html
- Select the option "Database Testing" from the drop-down menu
- Close the browser

ListBox:

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

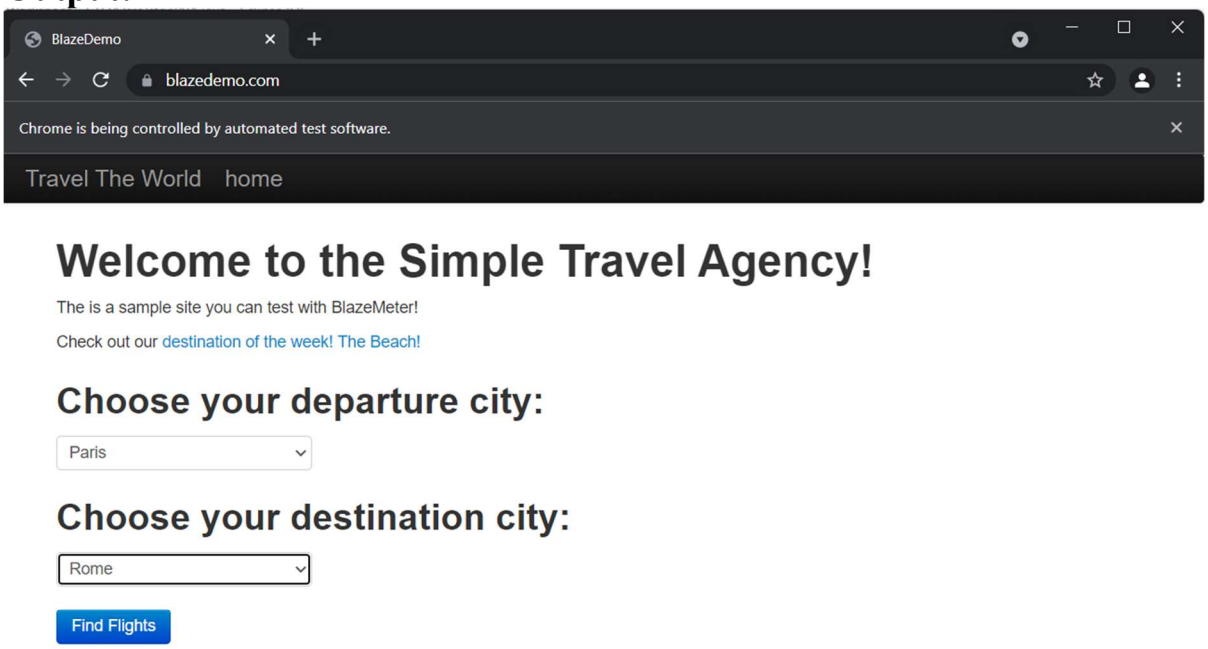
To identify the ListBox on webpage, look for select tag and attribute should be 'multiple' to select multiple items and there will be option tag which contains each item in it.

Code:

Drop Down

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class stqa09A {
	public static void main(String[] args) {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver wd = new ChromeDriver();
		wd.get("https://blazedemo.com/");
		Select s = new Select(wd.findElement(By.name("fromPort")));
		Select t = new Select(wd.findElement(By.name("toPort")));
		s.selectByVisibleText("Paris");
		t.selectByVisibleText("Rome");
	}
}
```

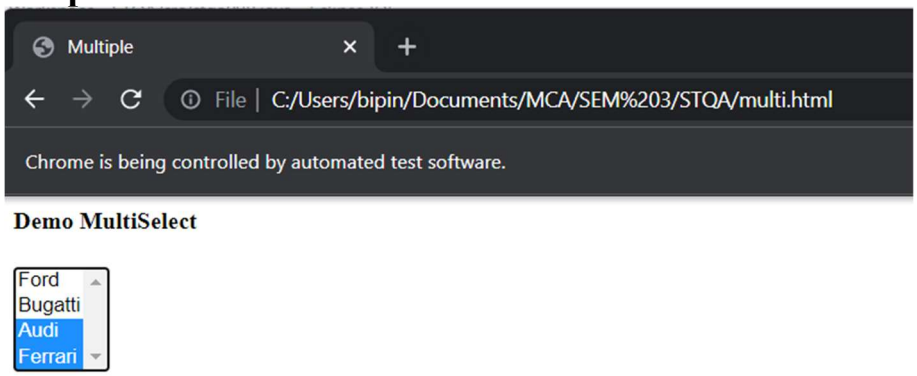
Output:



List Box

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class stqa09B {
	public static void main(String[] args) throws Exception {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver wd = new ChromeDriver();
		wd.get("file:///C:/Users/bipin/Documents/MCA/SEM
3/STQA/multi.html");
		Select s = new Select(wd.findElement(By.id("car")));
		if(s.isMultiple()) {
			s.selectByIndex(1);
			s.selectByValue("3");
			s.selectByVisibleText("Ferrari");
			Thread.sleep(2000);
			s.deselectByIndex(1);
			//s.deselectAll();
		}
	}
}
```

Output:



Conclusion:

Concept of DropDown and ListBox has been demonstrated.

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.

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

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.javatpoint.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.
The respective command that takes you forward by one page on the browser's history can be written as:

```
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
```

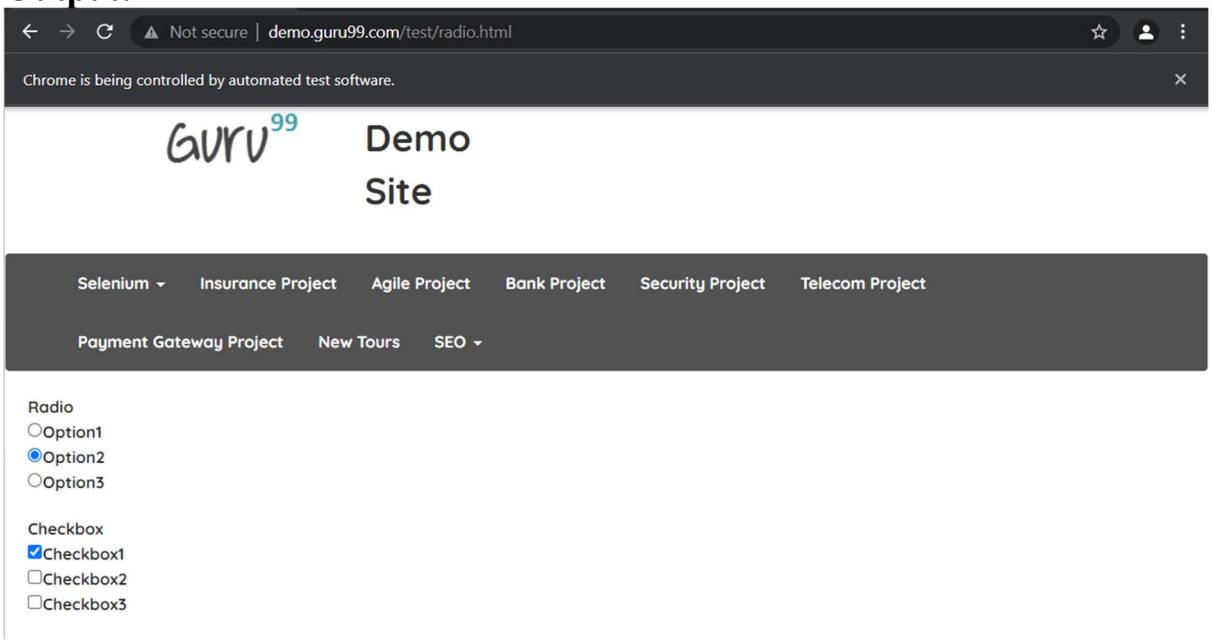
In WebDriver, this method refresh/reloads the current web page in the existing browser window. It neither accepts anything nor returns anything.
The respective command that takes you back by one page on the browser's history can be written as:

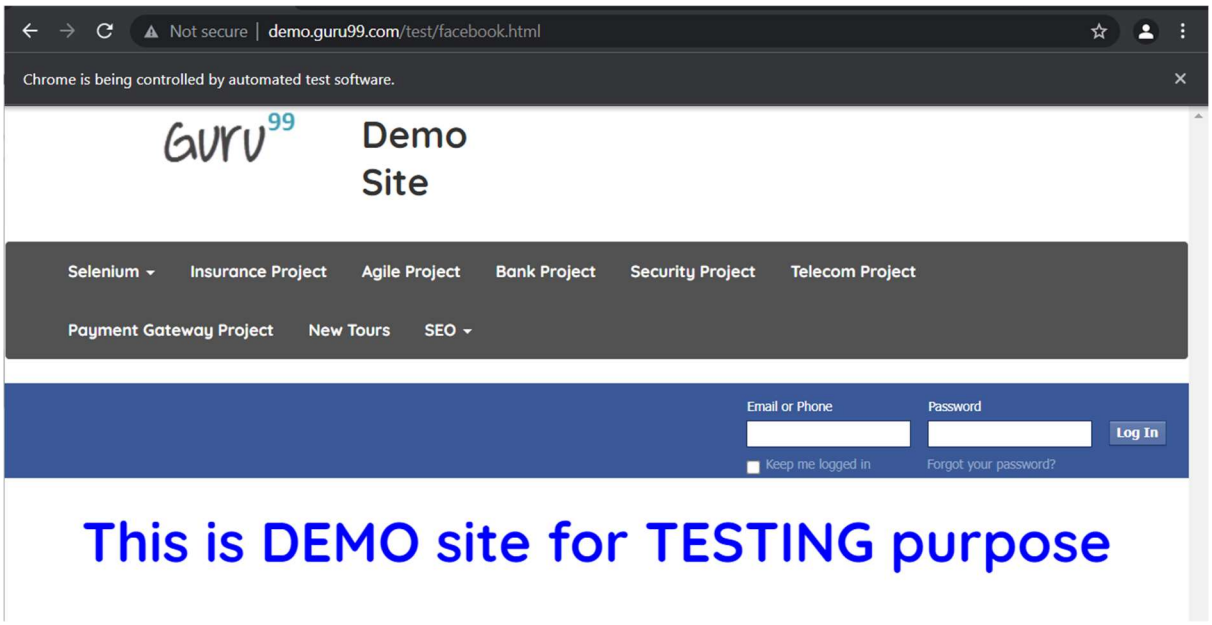
```
driver.navigate().refresh();
```

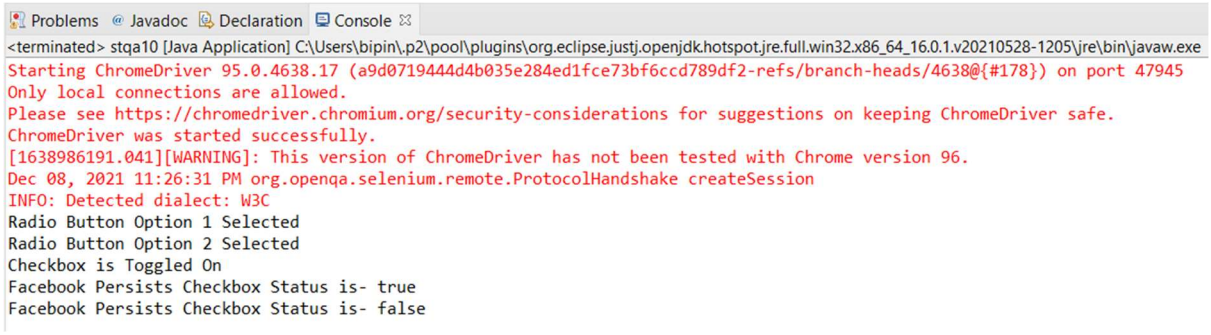
Code:

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

public class stqa10 {
	public static void main(String[] args) {

	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver driver = new ChromeDriver();
		driver.get("http://demo.guru99.com/test/radio.html");
		WebElement radio1 = driver.findElement(By.id("vfb-7-1"));
		WebElement radio2 = driver.findElement(By.id("vfb-7-2"));
		//Radio Button1 is selected
		radio1.click();
		System.out.println("Radio Button Option 1 Selected");
		//Radio Button1 is de-selected and Radio Button2 is selected
		radio2.click();
		System.out.println("Radio Button Option 2 Selected");
		// Selecting CheckBox
		WebElement option1 = driver.findElement(By.id("vfb-6-0"));
		// This will Toggle the Check box
		option1.click();
		// Check whether the Check box is toggled on
		if (option1.isSelected()) {
			System.out.println("Checkbox is Toggled On");
		} else {
			System.out.println("Checkbox is Toggled Off");
		}
		//Selecting Checkbox and using isSelected Method
		driver.get("http://demo.guru99.com/test/facebook.html");
		WebElement chkFBPersist =
				driver.findElement(By.id("persist_box"));
		for (int i=0; i<2; i++) {
			chkFBPersist.click ();
			System.out.println("Facebook Persists Checkbox Status is-
"+chkFBPersist.isSelected());
		}
	}
}
```

Output:







Conclusion:

Concept of CheckBox and Radio Button has been demonstrated.

Practical 11

Aim: Demonstrate action classes in Selenium.

Theory:

Actions class Selenium

Actions class includes a set of actions that a user performs on the web application using a Keyboard or a Mouse. It comes as a built-in feature of Selenium Webdriver for emulating advanced user interactions API. The interactions like clicking a button, entering a text in the search bar, drag-and-drop, and so on. These actions that are performed through Mouse and the keyboard; are automated using Selenium Actions.

Selenium Actions are the main elements that are responsible for the operations performed through these input devices. It is quite easy to understand how to use the Actions class in Selenium but the difficult part is to execute it. Ideally, the Selenium actions class which is the API for invoking action events, should be used instead of using input devices. A general form of code to perform actions class uses the following syntax:

```
Actions action = new Actions(driver); – To configure the action
action.moveToElement(element).click().perform(); -To click on the element
```

Different methods under the Actions class

There are two types of actions mainly performed in the web browser using Selenium, namely:

1. **Mouse Actions:**

There are several Mouse Actions provided by Actions Class, namely:

- click() – To click current location
- doubleClick() – To Perform double click on the mouse location
- clickAndHold() – To Perform mouse click without release.
- contextClick() – To perform right mouse click on the current mouse location
- moveToElement(WebElement target) – To move the mouse pointer to the centre of the targetlocation
- dragAndDrop(WebElement source, WebElement target) – To drag the element from the source and
- drop to the target location.
- dragAndDropBy(source, xOffset, yOffset) – To click and hold on current location then shifts by a
- given offset value and then release the mouse (X = Shift Horizontally, Y= Shift Vertically)
- release() – To release the left mouse button on current location

2. **Keyboard Actions:**

- sendKeys(sendKeys(WebElement target, java.lang.CharSequence… keys) : To send a series of keys to the element
- keyUp(WebElement target, java.lang.CharSequence key) : Performs key release after focusing on the target
- keyDown(WebElement target, java.lang.CharSequence key): To perform key press without release.

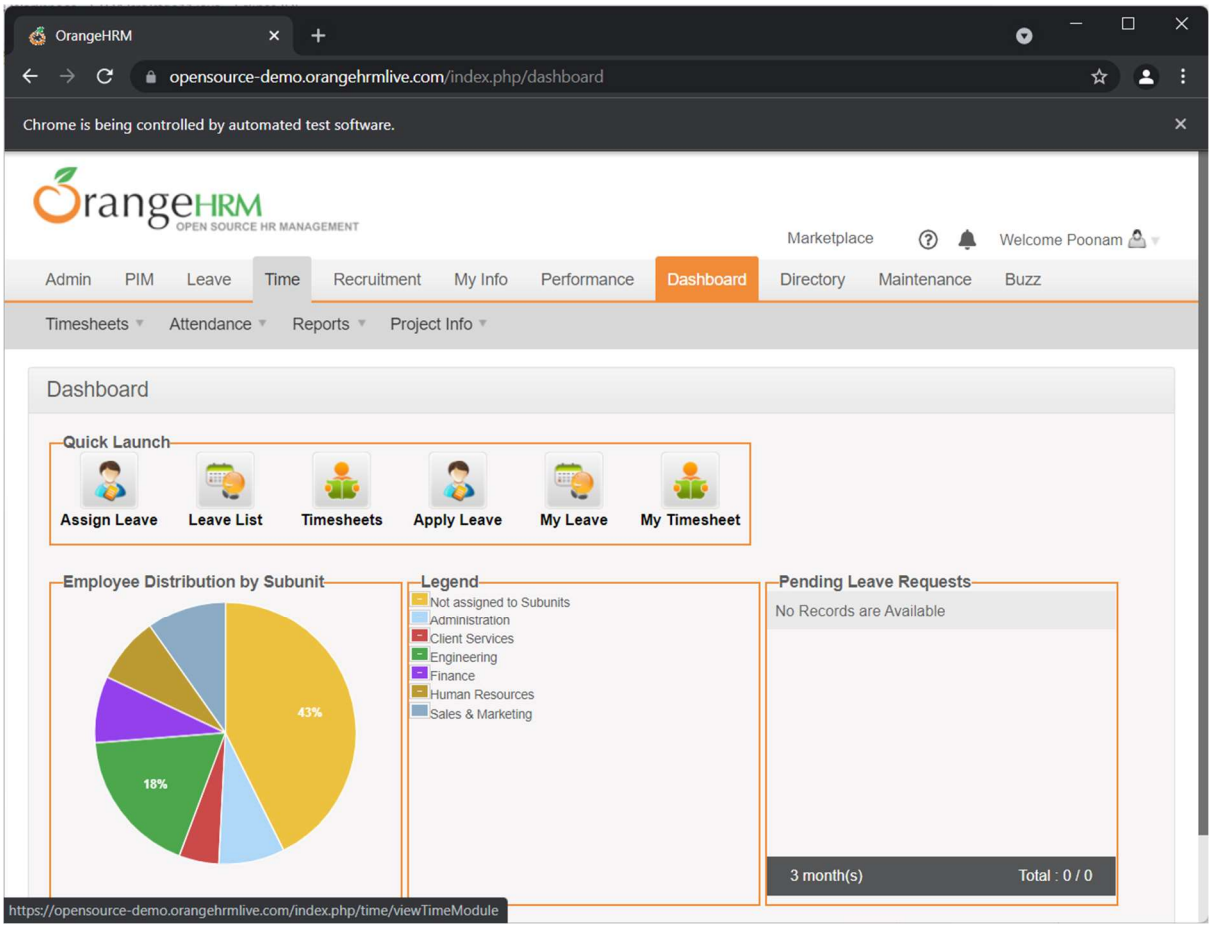
Code:

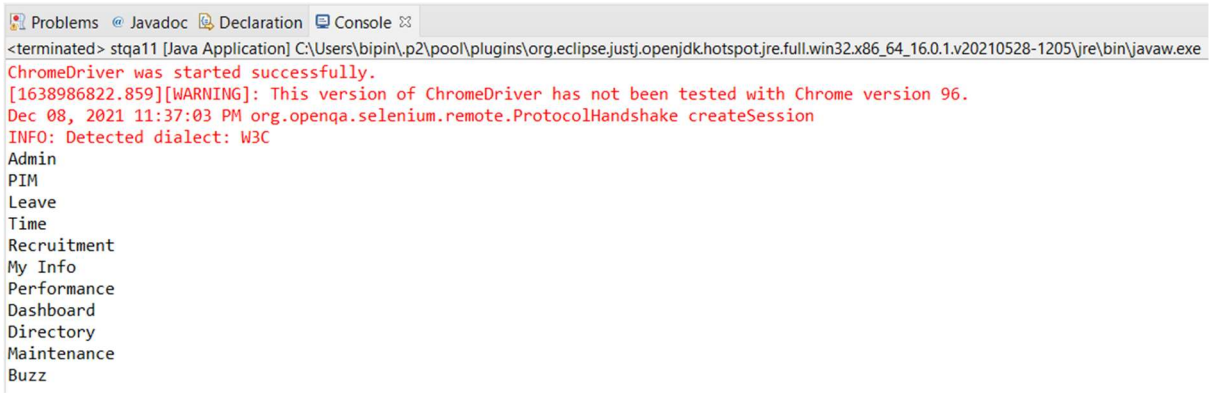
```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;

public class stqa11 {
	public static void main(String[] args) throws Exception {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		WebDriver wd = new ChromeDriver();
		wd.get("https://opensource-demo.orangehrmlive.com/");
		wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
		wd.findElement(By.id("txtUsername")).sendKeys("admin");//locator id

	wd.findElement(By.name("txtPassword")).sendKeys("admin123");//locater name
		wd.findElement(By.className("button")).click();//locator className
		Actions act = new Actions(wd);
	//act.moveToElement(wd.findElement(By.className("firstLevelMenu"))).perform
();
	//act.moveToElement(wd.findElement(By.id("menu_recruitment_viewRecruitmentM
odule"))).perform();
		List<WebElement>
menu=wd.findElements(By.className("firstLevelMenu"));
		for(int i=0;i<=menu.size()-1;i++)
		{
			System.out.println(menu.get(i).getText());//print text of all
the element on console
			act.moveToElement(menu.get(i)).perform();//to perform mouse
hover on all elements of list
			Thread.sleep(2000);
		}
		wd.findElement(By.partialLinkText("Welcome")).click();//locator
partiallinkText
		wd.findElement(By.linkText("Logout")).click();//locator linkText
		wd.close();
	}
}
```

Output:

Log In

Actions





Conclusion:

Concept of Action Classes has been demonstrated.

Practical 12

Aim: Installation of TestNg , running TestNg and TestNg annotations.

Theory:

What is TestNG?

- TestNG is a very important framework when you are actually developing the framework from scratch level.
- TestNG provides you full control over the test cases and the execution of the test cases. Due to this reason, TestNG is also known as a testing framework.
- Cedric Beust is the developer of a TestNG framework.
- If you want to run a test case A before that as a pre-request you need to run multiple test cases before you begin a test case A. You can set and map with the help of TestNG so that pre-request test cases run first and then only it will trigger a test case A. In such way, you can control the test cases.
- TestNG framework came after Junit, and TestNG framework adds more powerful functionality and easier to use.
- It is an open source automated TestNG framework. In TestNG, NG stands for "**Next Generation**".
- TestNG framework eliminates the limitations of the older framework by providing more powerful and flexible test cases with help of easy annotations, grouping, sequencing and parametrizing.

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

1. 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 is failed. Then correct the errors present in the fourth method, now you want torun only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.

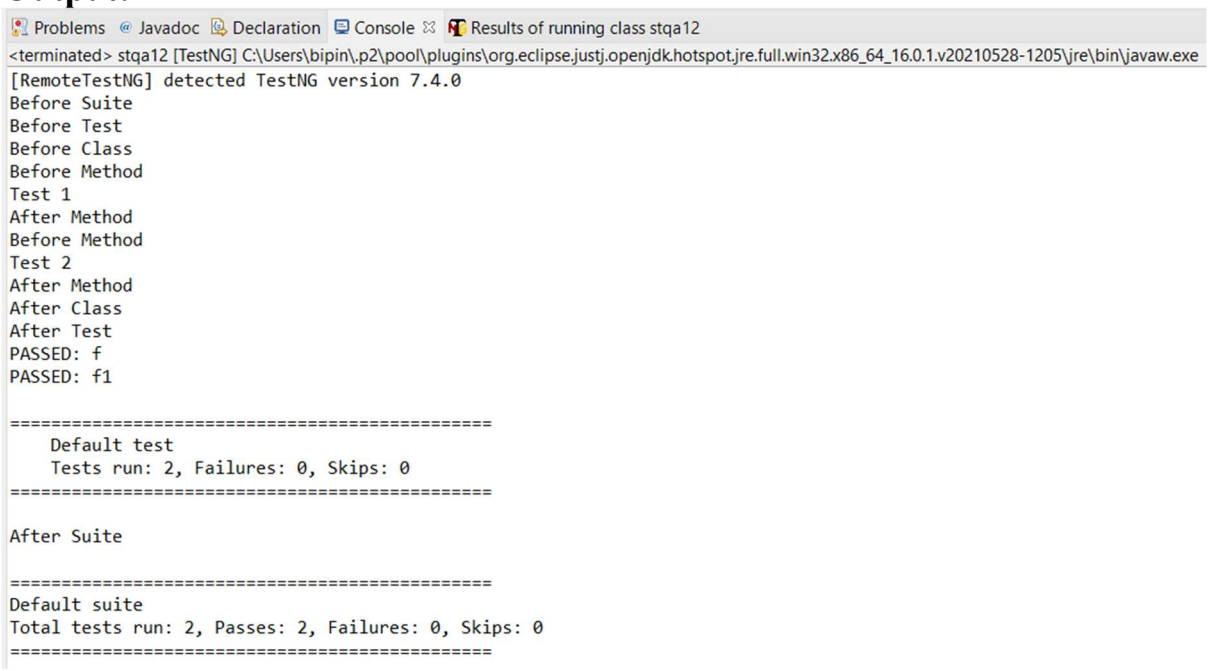
2. The TestNG in Selenium provides an option, i.e., testng-failed.xml filein test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

Beside 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 executethrough the eclipse as well as from the command line.

Code:

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

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:



Conclusion:

Concept of TestNg has been demonstrated.

Practical 13

Aim: Demonstrate data driven Framework.

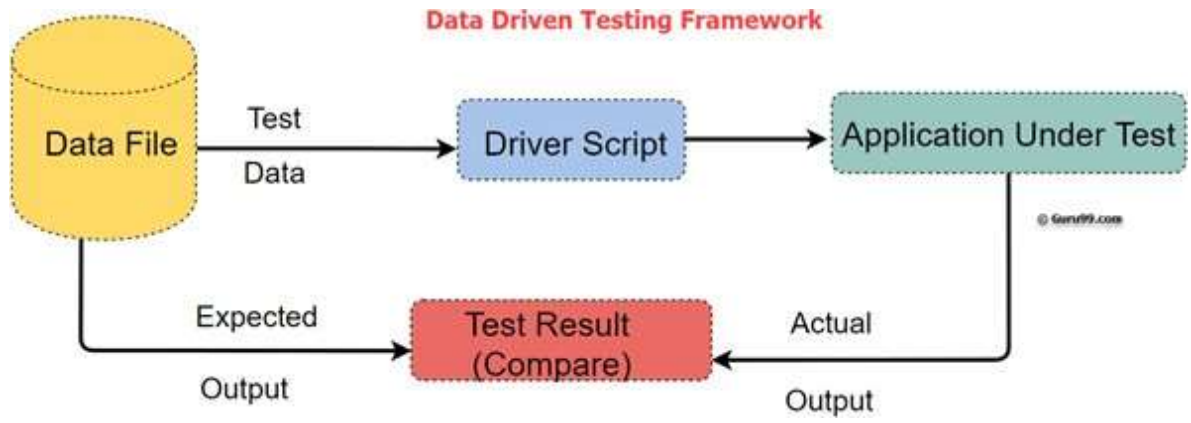
Theory:

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 toinput a single test script that can execute tests for all test data from a tableand 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 inputvalues 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.



Code:

```
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 stqa13 {
	WebDriver driver;
	@Test(dataProvider="testdata")
	public void demoClass(String username, String password) throws
	InterruptedException {
	System.setProperty("webdriver.chrome.driver","C:\\Users\\bipin\\Documents\\
MCA\\SEM 3\\STQA\\ChromeDriver\\chromedriver.exe");
		driver = new ChromeDriver();
		driver.get("https://www.phptravels.net/login");
		driver.findElement(By.name("email")).sendKeys(username);
		driver.findElement(By.name("password")).sendKeys(password);
```

```
	driver.findElement(By.xpath("/html/body/div[1]/div/div[2]/div[2]/div/form/d
iv[3]/button")).click();
		Thread.sleep(5000);
		Assert.assertTrue(driver.getTitle().matches("Dashboard -
PHPTRAVELS"), "Invalid credentials");
		System.out.println("Login successful");
	}

	@AfterMethod
	void ProgramTermination() {
		driver.quit();
	}

	@DataProvider(name="testdata")
	public Object[][] testDataExample(){
		ReadExcelFile configuration = new
ReadExcelFile("C:\\Users\\bipin\\Documents\\MCA\\SEM 3\\STQA\\XYZ.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;
	}
}
```

ReadExcelFile.java

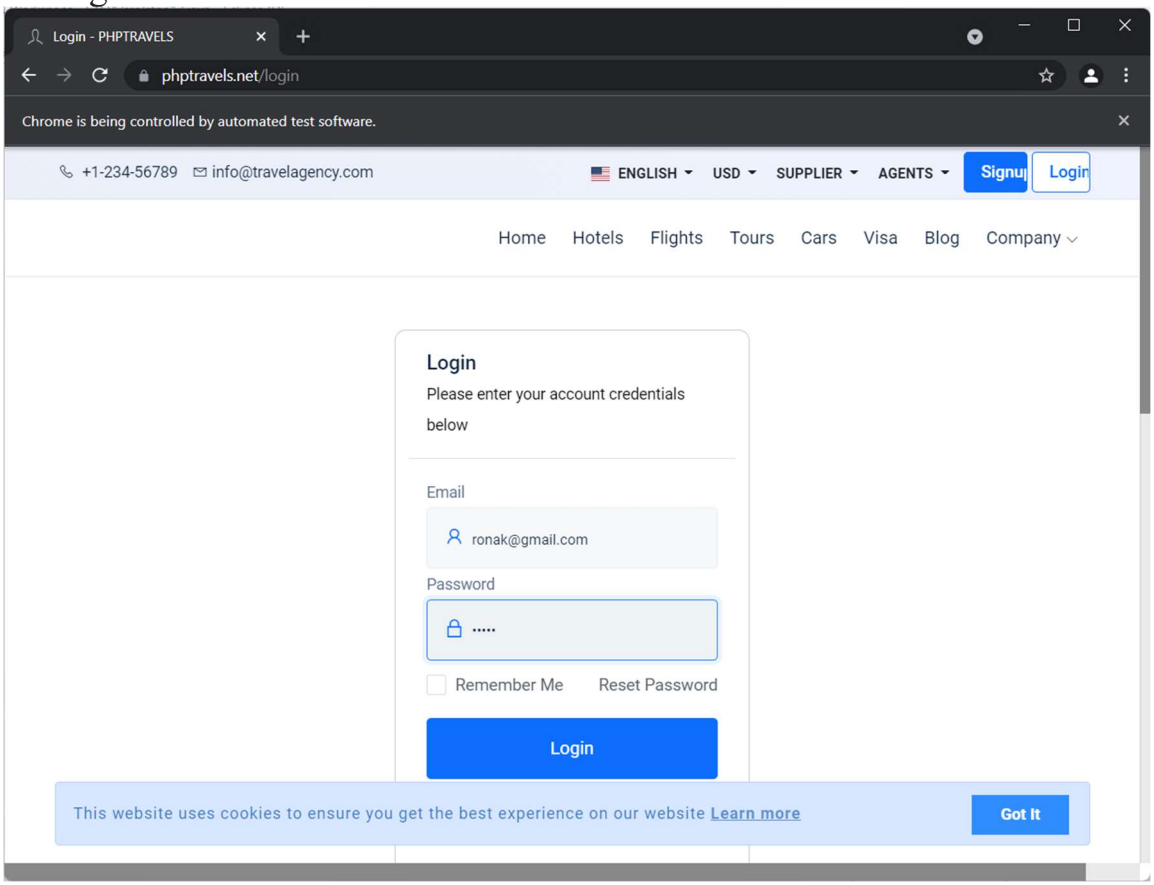
```
import java.io.File;
import java.io.FileInputStream;
import org.apache.poi.xssf.usermodel.XSSFSheet;
import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ReadExcelFile{
	XSSFWorkbook work_book;
	XSSFSheet sheet;
	public ReadExcelFile(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:

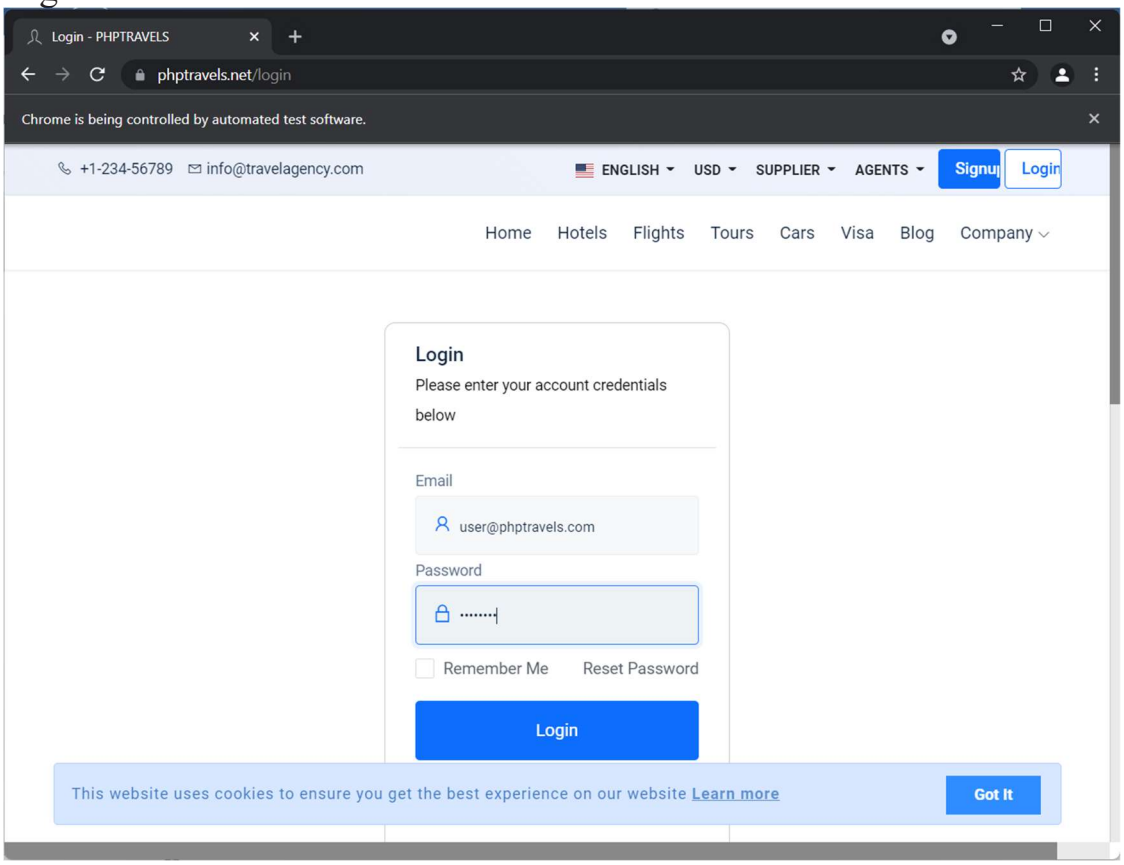
Data in excel sheet:

	A	B
1	ronak@gmail.com	ronak
2	anirudha@gmail.com	xyz
3	user@phptravels.com	demouser
4		

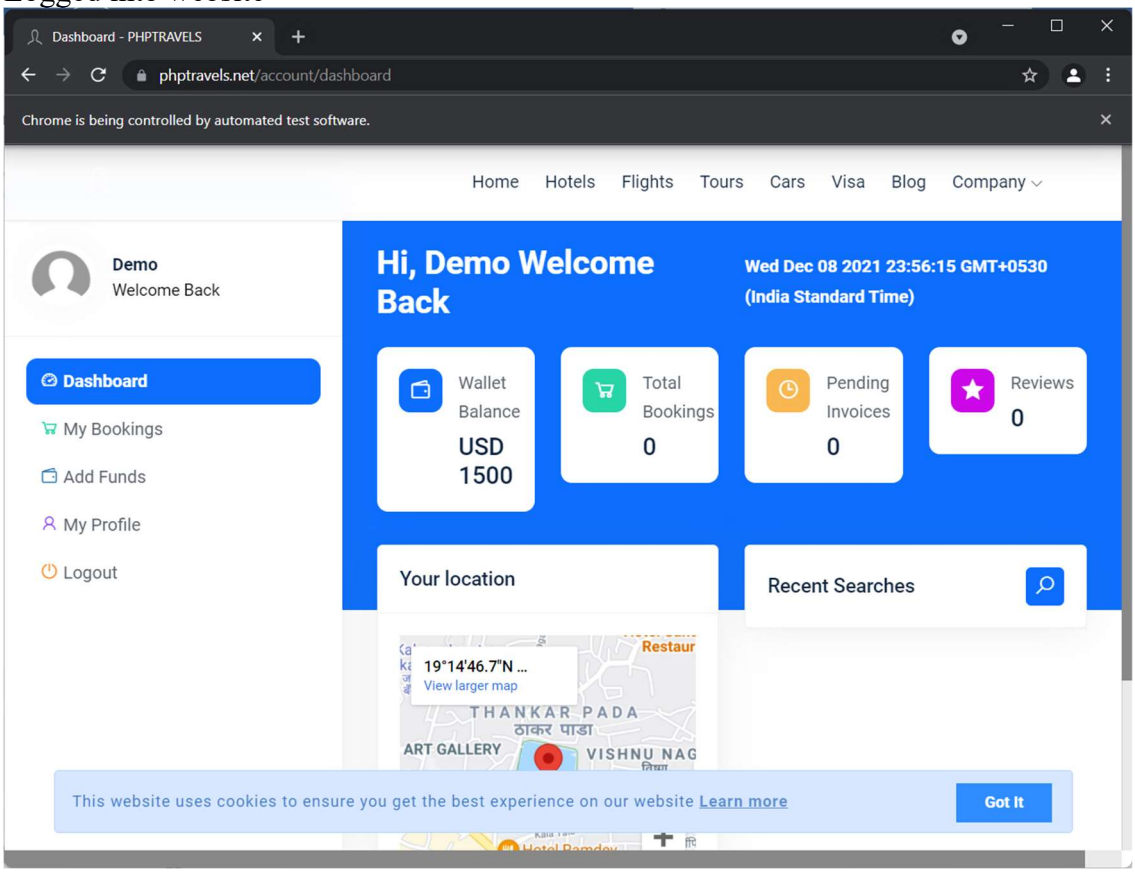
Wrong Credentials



Right Credentials



Logged into website



Console:

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


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

Conclusion:

Concept of Data Driven Framework has been demonstrated.