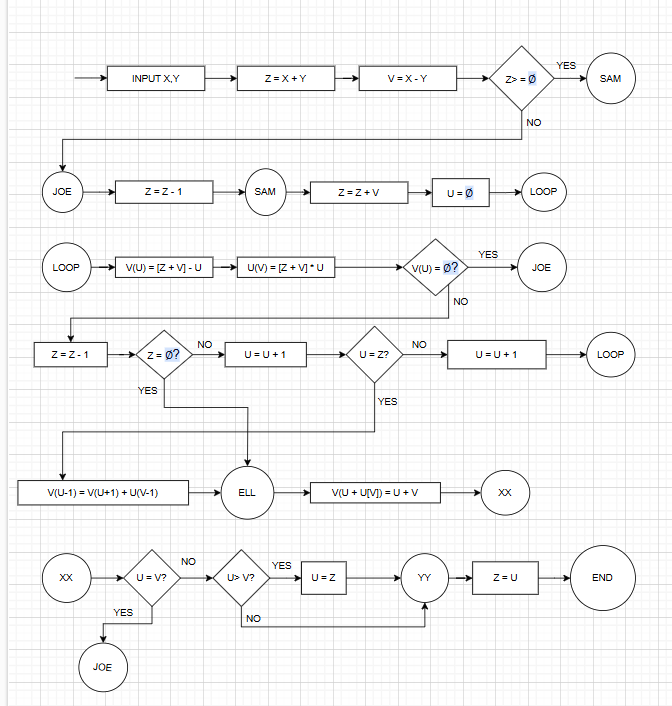
PRACTICAL-1

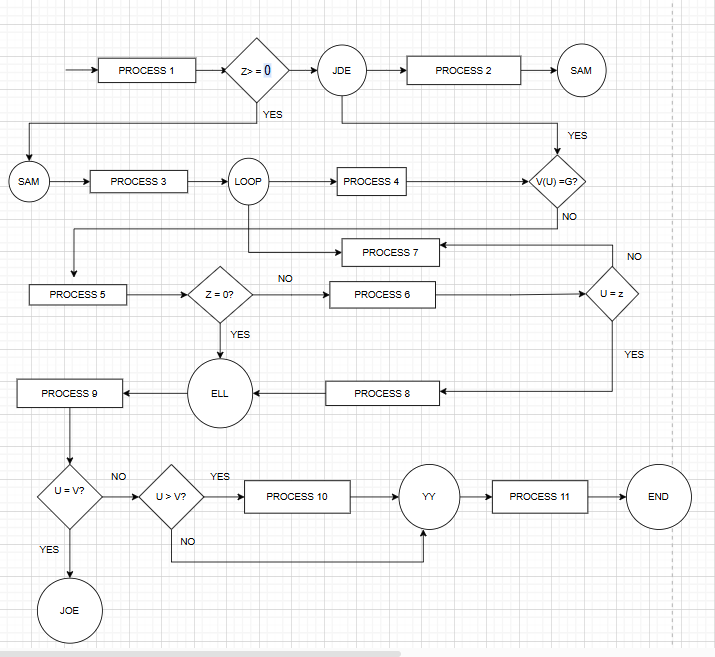
Aim: Flow graphs and Path Testing using Some elements

Output:

one-to-one flowchart :



Control Flowgraph :



Practical :- 2

Aim: - Installation of Selenium IDE & case test using Selenium IDE

To set up Selenium with TestNG, Eclipse, and Chrome Driver for automation testing in Java, follow these steps:

Steps:

Install Java Development kit (JDK)

Ensure that you have the JDK installed on your system. You can download it from the [Oracle website](https://www.oracle.com/java/technologies/javase-jdk11-downloads.html) or install it using a package manager like apt-get for Linux.

To verify if JDK is installed, open the terminal (Command Prompt for Windows) and run:

Java -version

Install Eclipse IDE

Download Eclipse: Go to the Eclipse Downloads page and download the "Eclipse IDE for Java Developers".

Install Eclipse: Follow the installation instructions based on your operating system.

After installing Eclipse Create a new project -> Mavan Project -> Add Group Id and Artifact ID and create a simple mavan project.

To set up selenium go to Mavan Repository(https://mvnrepository.com/) and search for selenium java -> copy the dependency for version 3.141.5(<https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java/3.141.5>) and paste it in the pom.xml file of your mavan project.

To set up testng go to Mavan Repository(https://mvnrepository.com/) and search for testng -> copy the dependency for version 6.14.3(https://mvnrepository.com/artifact/org.testng/testng) and paste it in the pom.xml file of your mavan project.

Install TestNG Plugin for Eclipse:

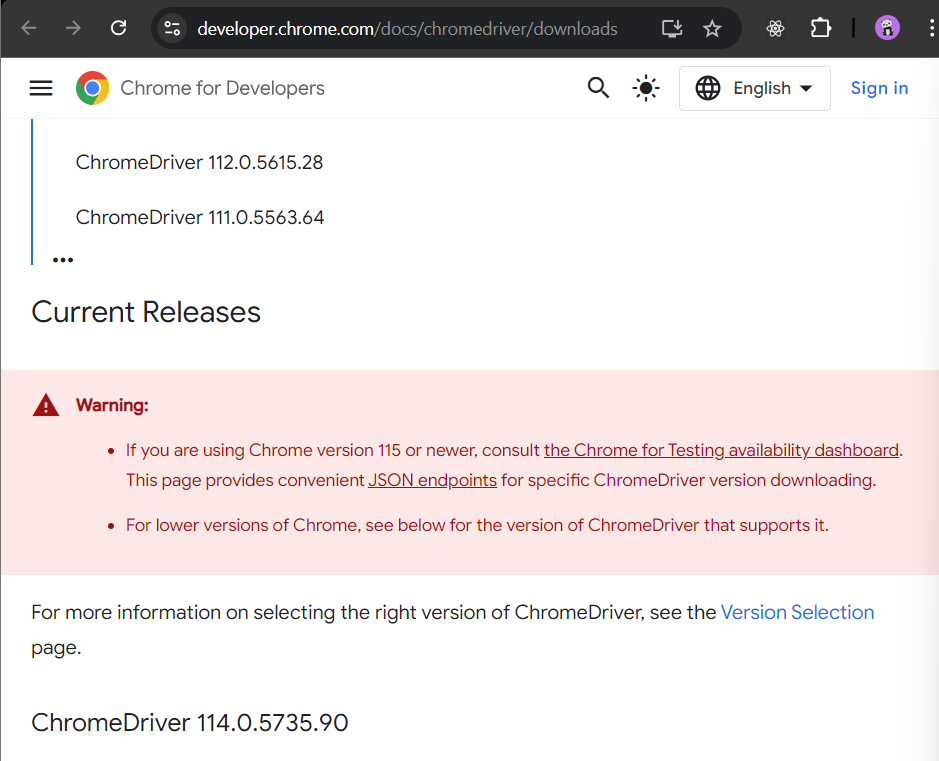
Go to Help > Eclipse Marketplace.

Search for "TestNG" and click Go.

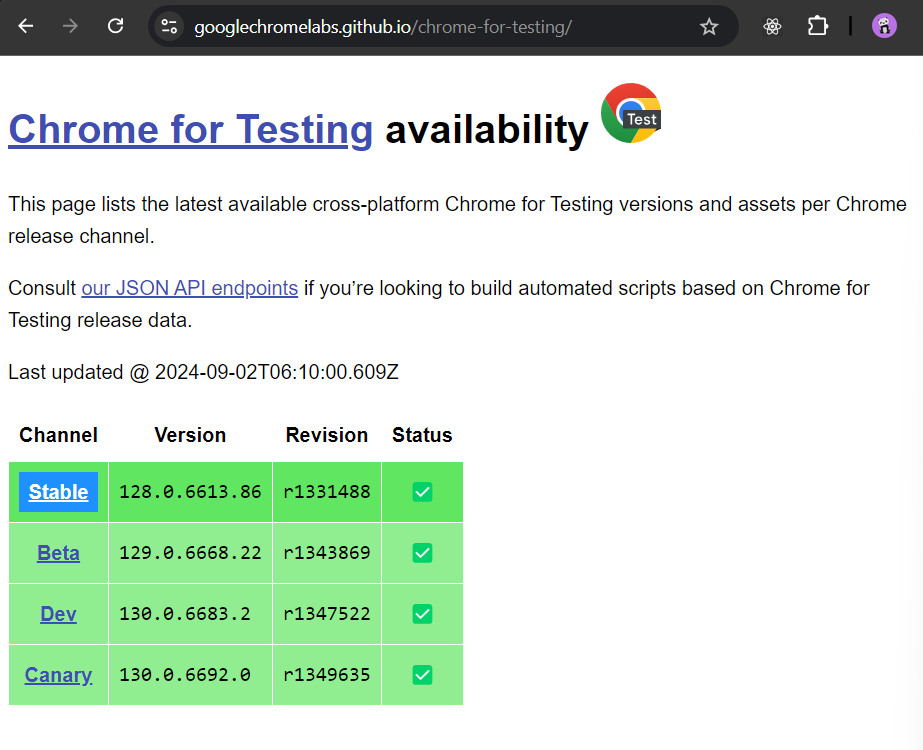
Install the TestNG plugin and restart Eclipse.

Install Chrome Driver

Search for chrome driver -> Click on chrome for testing availability dashboard.



Click on stable



Copy the URL according to your device and paste it in the browser to download it

After the file is downloaded Extract the file and copy the chromedriver.exe file. -> open eclipse creates a new folder with name “drivers” and paste the copied chromedriver.exe file in it.

Create a Package and class in the project -> add a new function -> add a system.out. println(“Hello World”);

And the test it as Run as -> Testng Test. Check your console for the output.

package demoselenium;

import org.testng.annotations.Test;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.WebDriver;

public class testdemo {

*@Test*

public void myTest()

{

System.***out***.println("hello World");

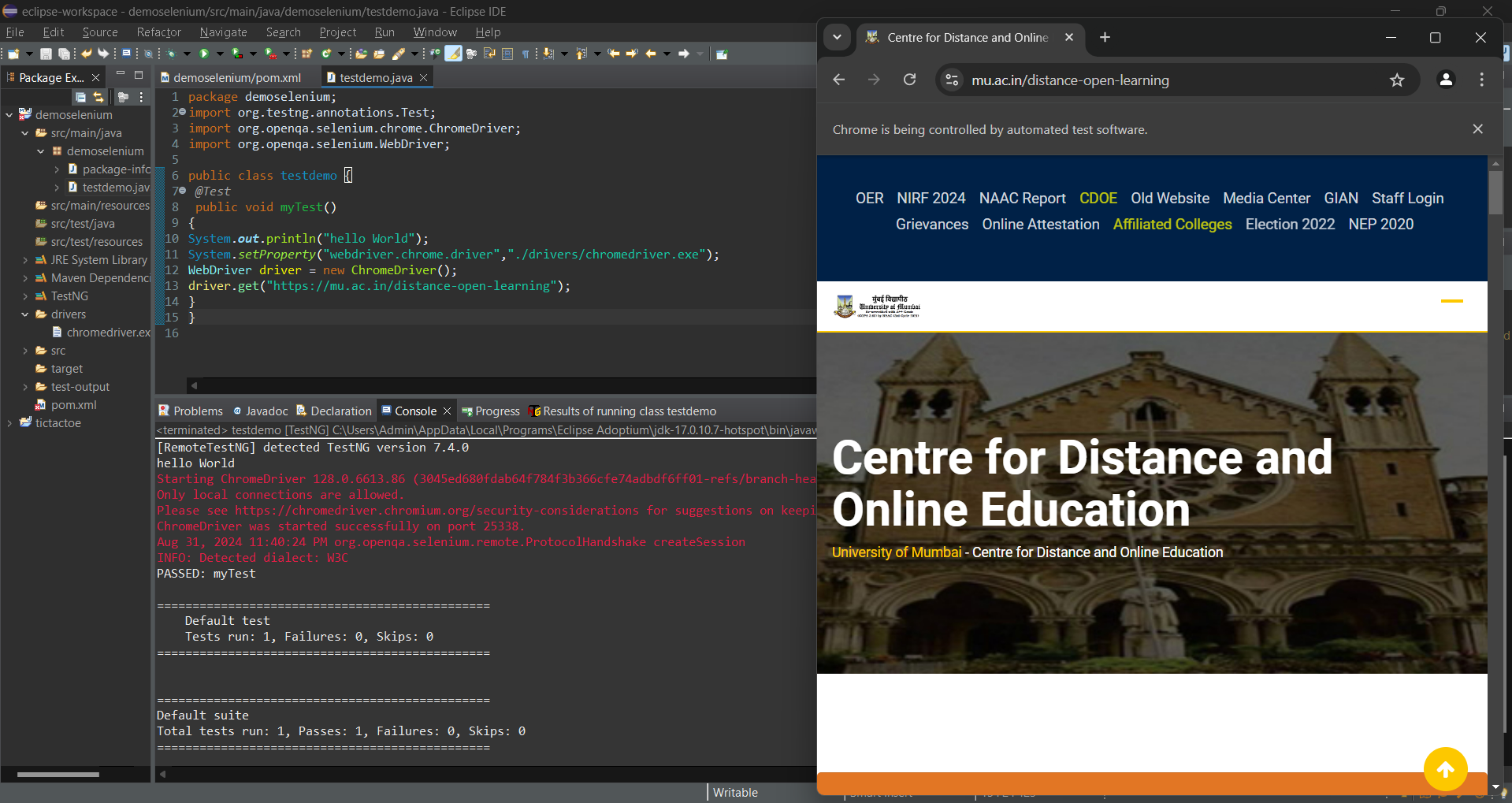
System.*setProperty*("webdriver.chrome.driver","./drivers/chromedriver.exe");

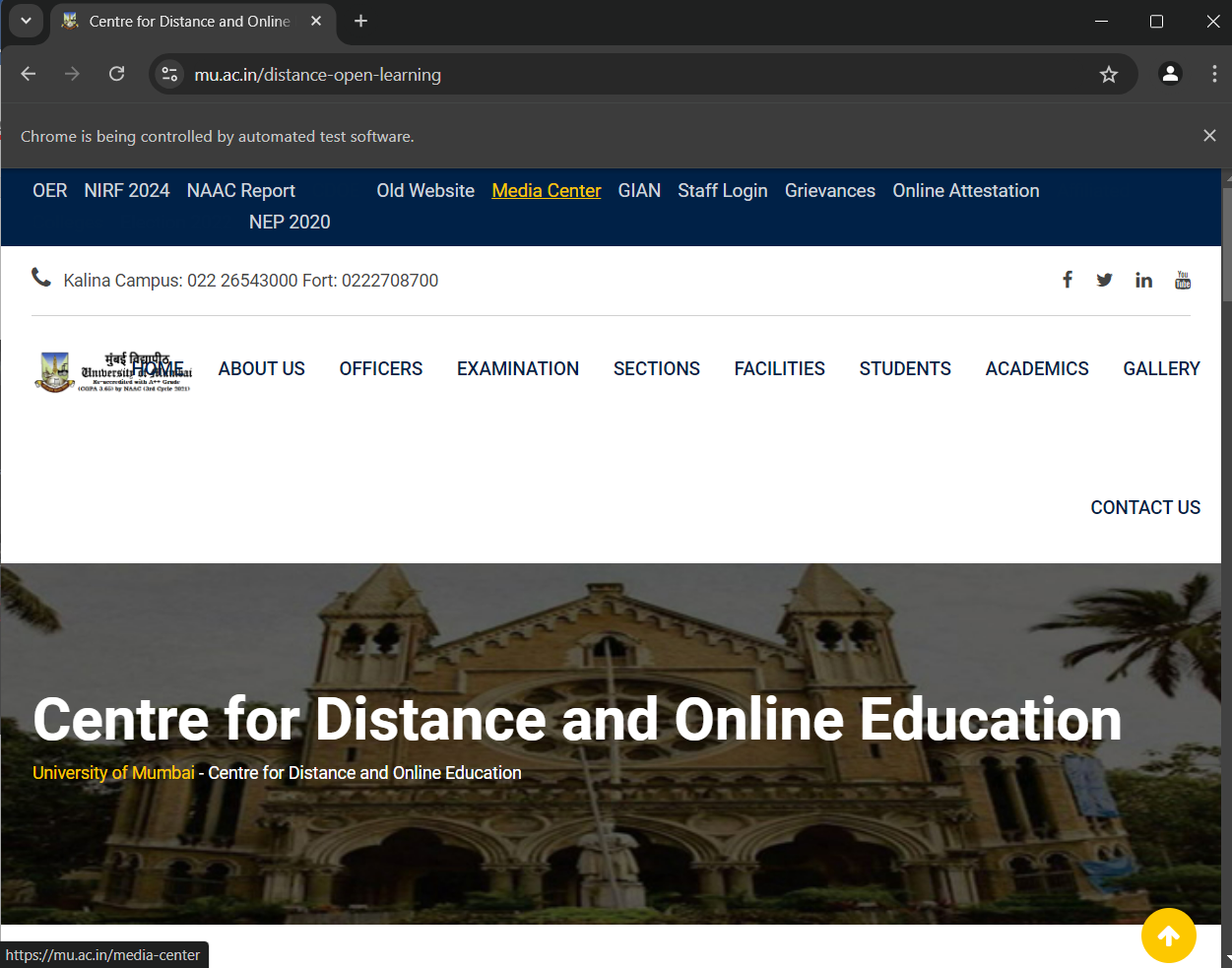
WebDriver driver = new ChromeDriver();

driver.get("https://mu.ac.in/distance-open-learning");

}}

OUTPUT:-





Practical 3

Aim: Implementing handling multiple frames

CODE:-

package demoselenium;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

import org.testng.annotations.Test;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

public class testdemo {

*@Test*

public void myTest()

{

//System.out.println("hello World");

System.*setProperty*("webdriver.chrome.driver","./drivers/chromedriver.exe");

WebDriver driver = new ChromeDriver();

driver.get("https:kjsit.somaiya.edu.in/en/");

System.***out***.println("Parent:"+driver.getTitle());

String parentwindow = driver.getWindowHandle();

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

driver.findElement(By.*linkText*("B.Tech.(IT)")).click();

Set<String>Windowhandles = driver.getWindowHandles();

List<String>Windows = new ArrayList<String>(Windowhandles);

driver.switchTo().window(Windows.get(1));

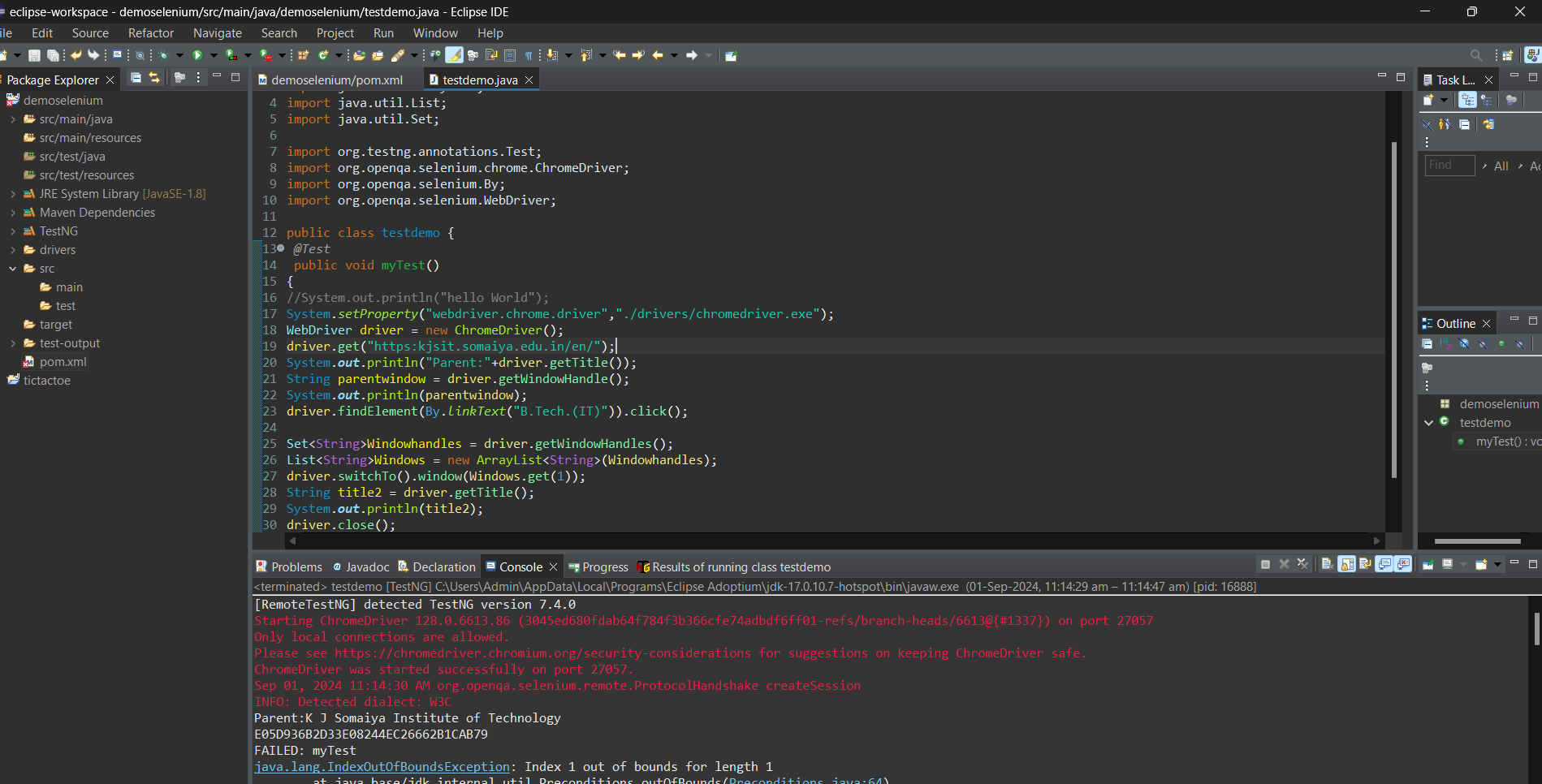
String title2 = driver.getTitle();

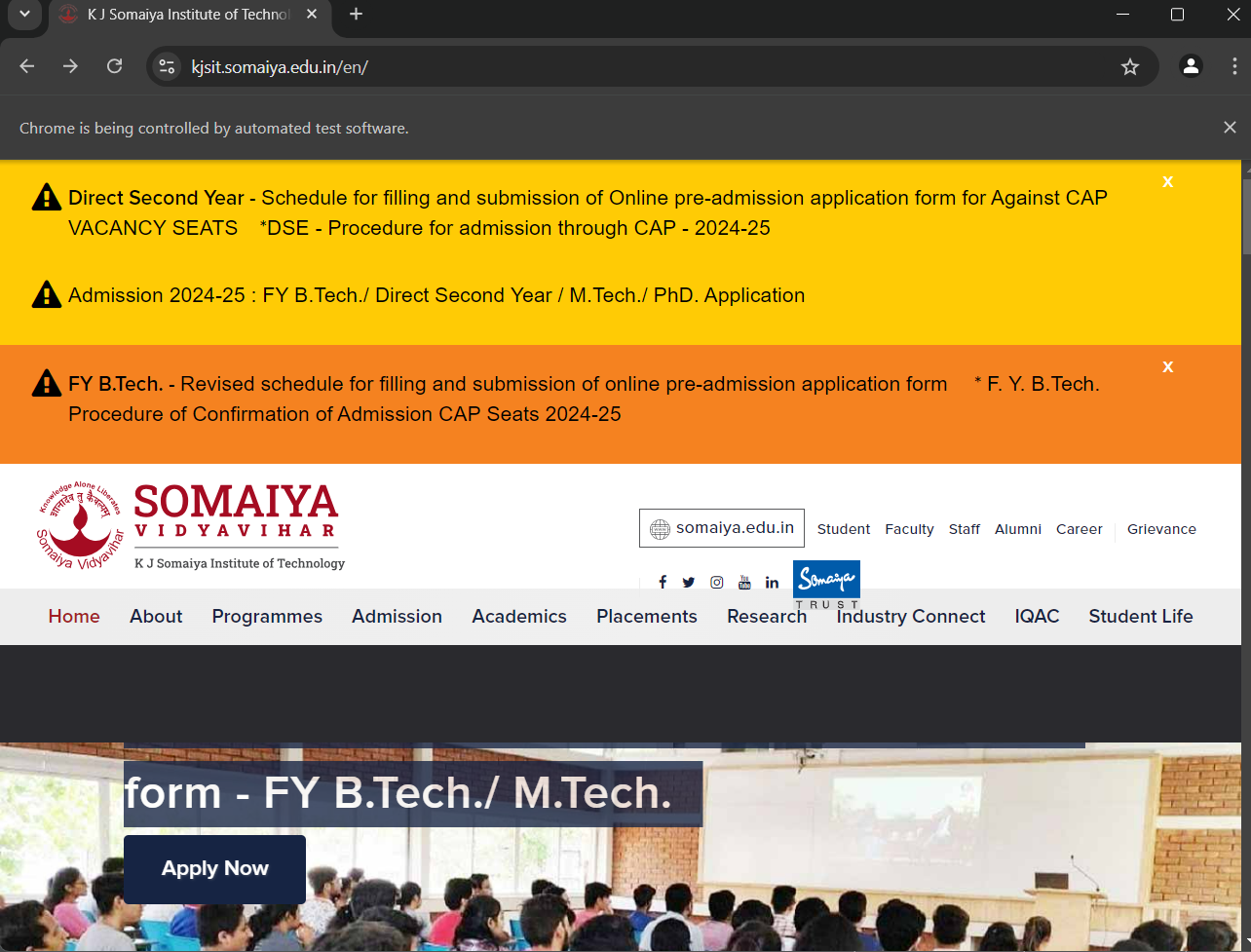
System.***out***.println(title2);

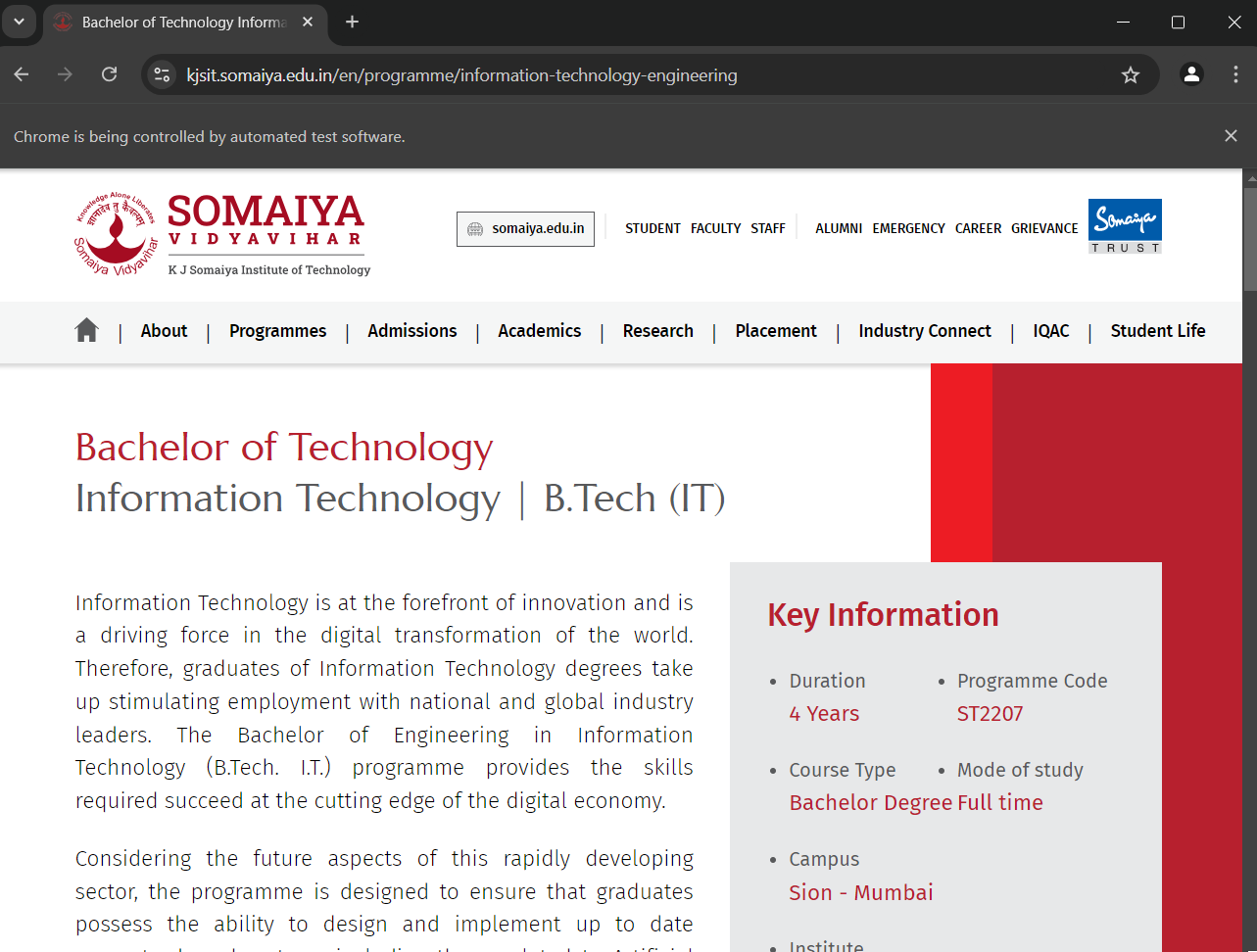
driver.close();

}}

OUTPUT:-







Practical 4

Aim: Implementing Selenium WebDriver- find element command, Locator(id,css selector, Xpath), Input Box, Buttons, Submit Buttons.

CODE:-

package demoselenium;

//import java.util.ArrayList;

//import java.util.List;

//import java.util.Set;

//import org.testng.annotations.Test;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.By;

//import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

public class testdemo {

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

System.*setProperty*("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

driver.get("https://letcode.in/frame");

//frame

//driver.switchTo().frame("firstFr");

//driver.switchTo().frame(0);

WebElement frameEle = driver.findElementByXPath("//\*[@id=\"firstFr\"]");

driver.switchTo().frame(frameEle);

driver.findElement(By.*name*("fname")).sendKeys("IDOL");

driver.findElement(By.*name*("lname")).sendKeys("MCA");

Thread.*sleep*(2000);

//nested frame

WebElement innerFrames = driver.findElement(By.*cssSelector*("iframe.has-background-white"));

driver.switchTo().frame(innerFrames);

driver.findElement(By.*name*("email")).sendKeys("idol.mca@somaiya.edu");

Thread.*sleep*(2000);

//parent Frame

driver.switchTo().parentFrame();

driver.findElement(By.*name*("fname")).sendKeys(" SY ");

Thread.*sleep*(2000);

//default frame

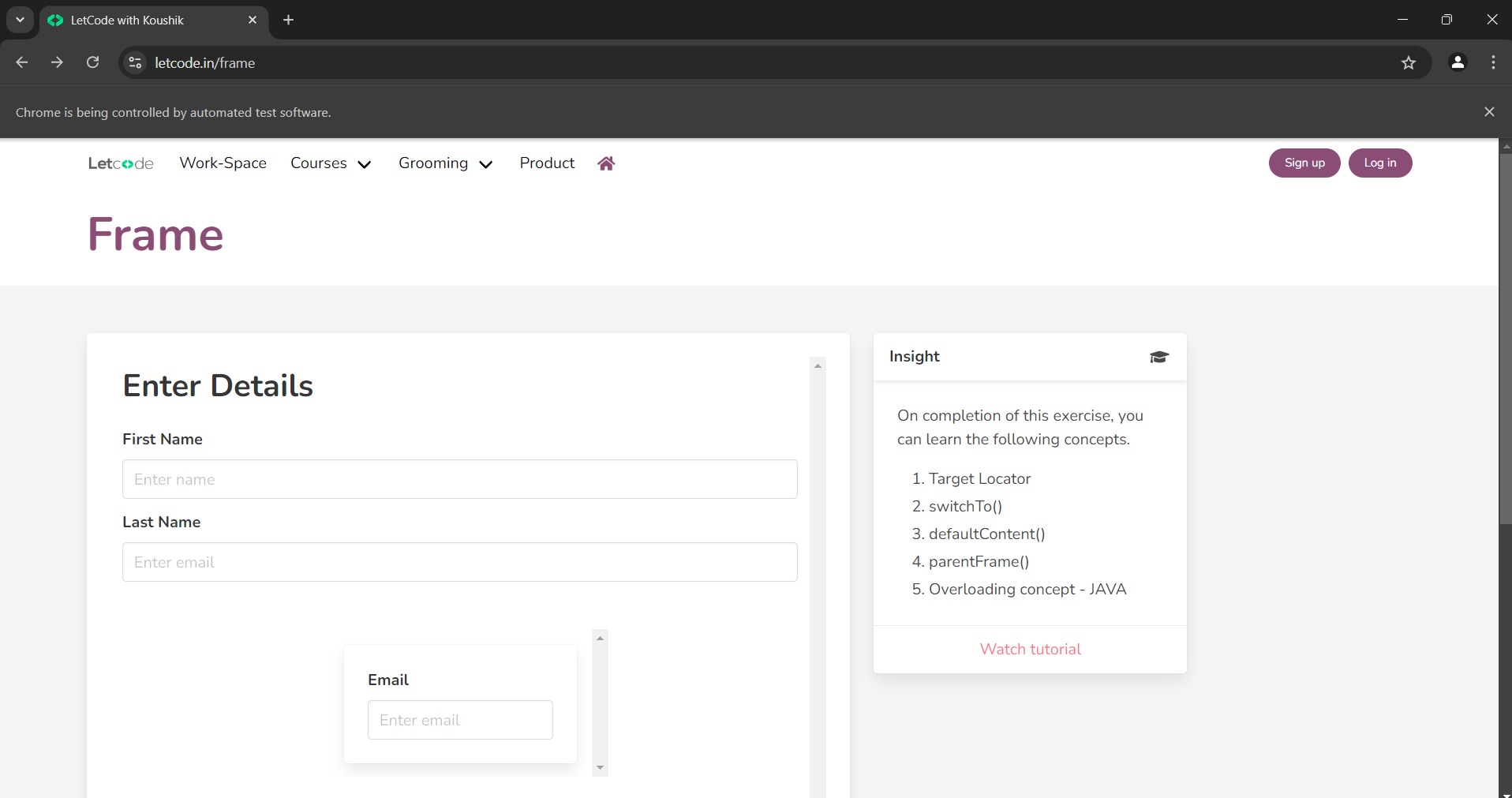
driver.switchTo().defaultContent();

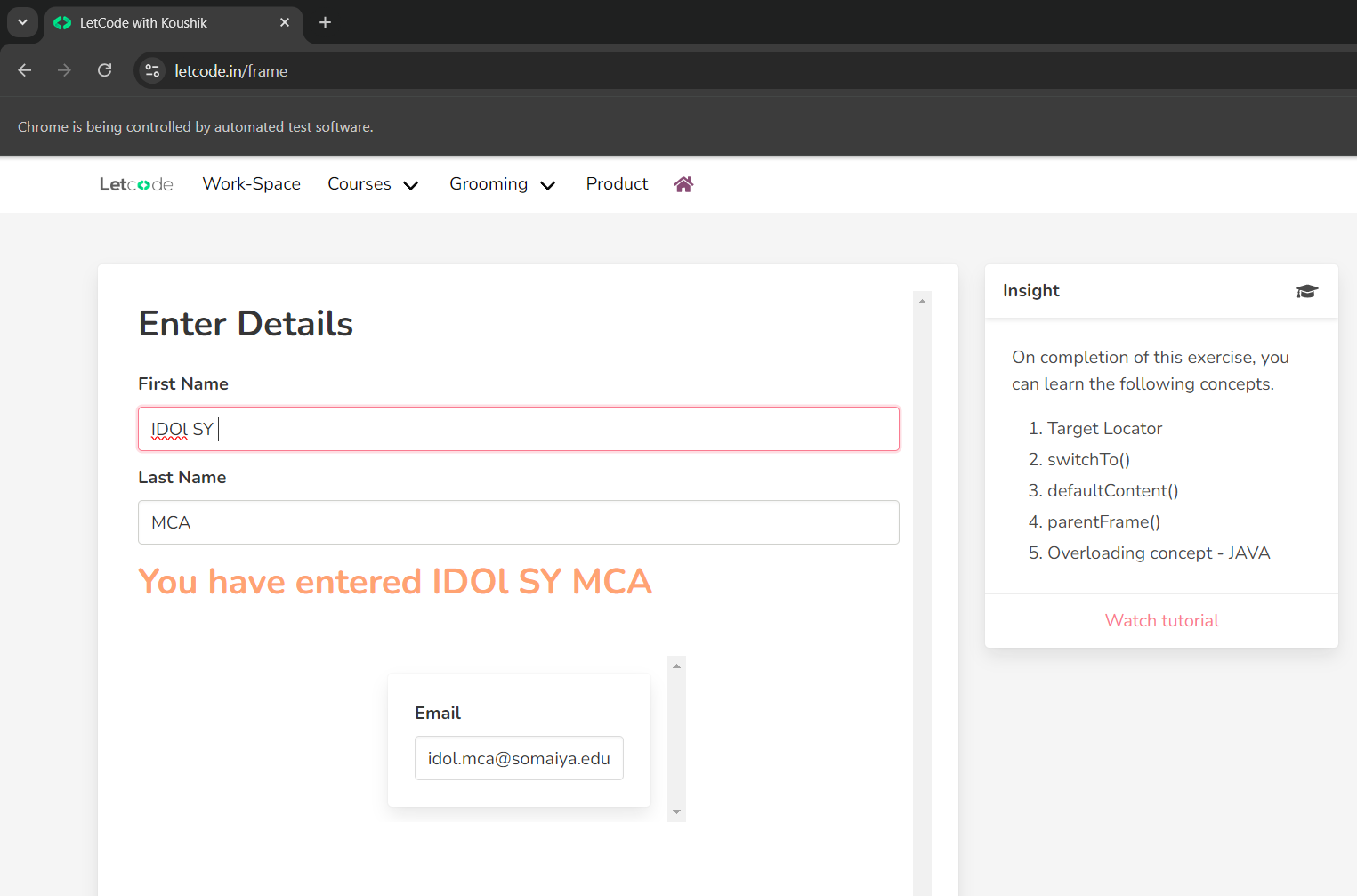
driver.findElement(By.*linkText*("Log in")).click();

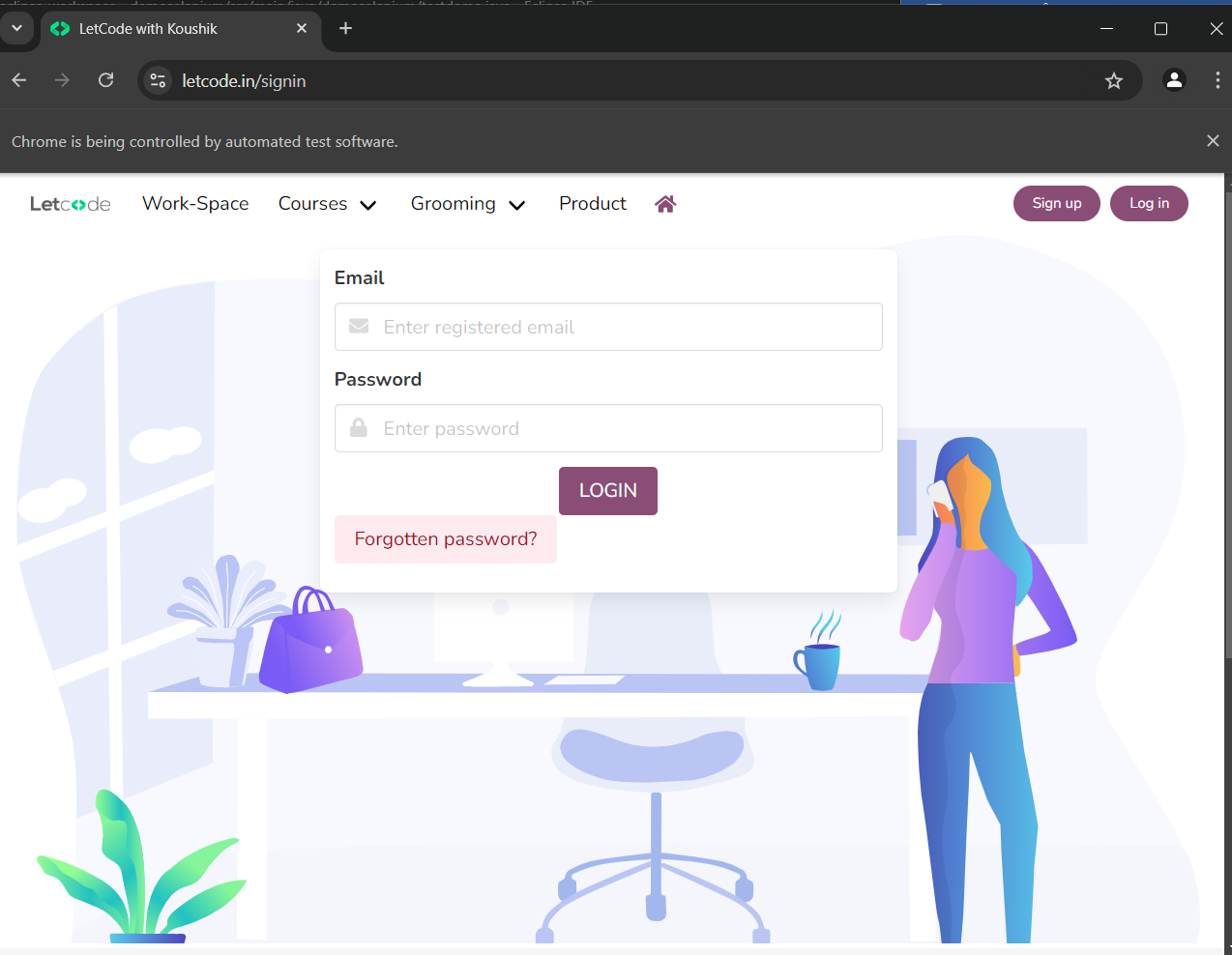
Thread.*sleep*(2000);

// driver.quit(); }}

OUTPUT:-







Practical 5

Aim: Implementing methods in TestNG file.

CODE:-

package demoselenium;

//import java.util.ArrayList;

//import java.util.List;

//import java.util.Set;

//import org.testng.annotations.Test;

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

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;

//import org.openqa.selenium.By;

//import org.openqa.selenium.WebDriver;

//import org.openqa.selenium.WebElement;

public class testdemo {

@Test

public void f() {

System.out.println("First Test");

}

@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

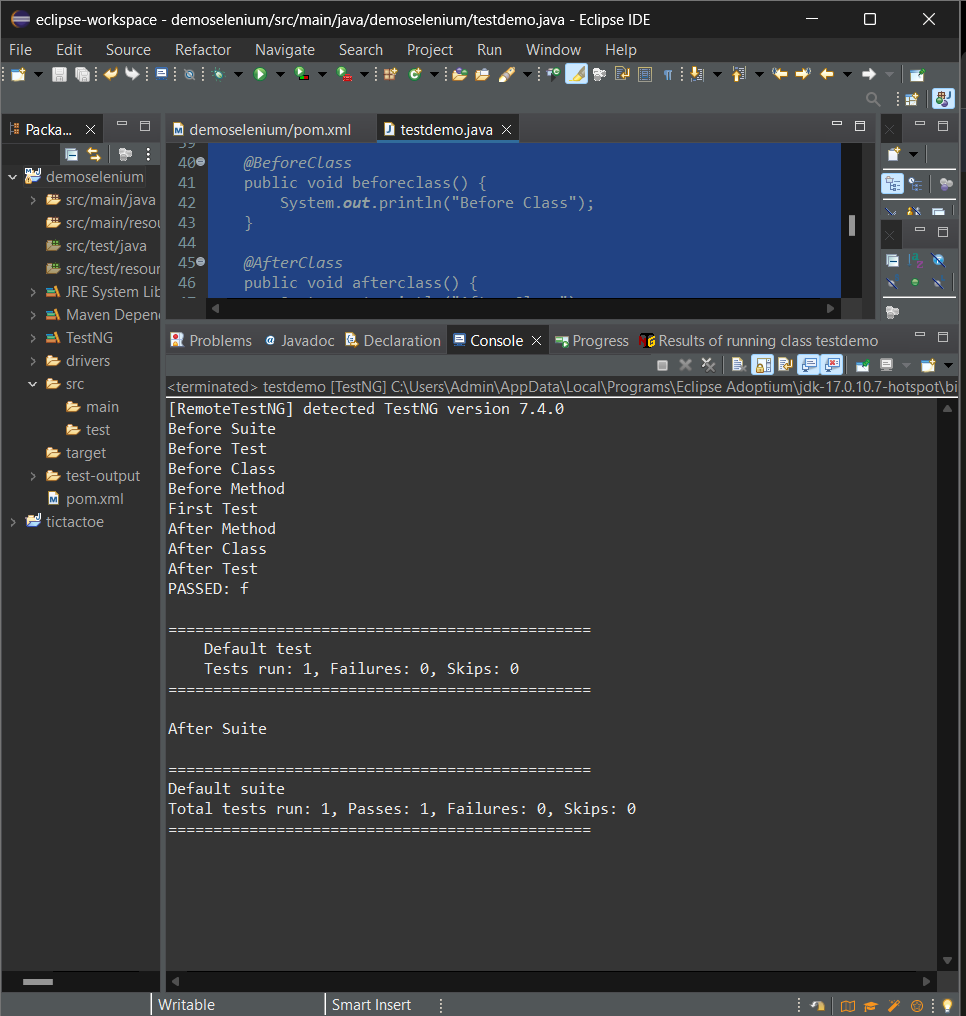
public void afterSuite()

{

System.out.println("After Suite");

}}

OUTPUT:-



Practical 6

Aim: Select Value from DropDown using Selenium Webdriver.

CODE:-

package lamdaDemo;

import org.openqa.selenium.WebDriver.Options;

import org.openqa.selenium.WebDriver.Window;

import org.openqa.selenium.chrome.ChromeDriver;

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

public class dropdown {

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

System.setProperty("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

Options manage = driver.manage();

Window = manage.window();

window.maximize();

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

Thread.sleep(1000);

Select departure = new Select(driver.findElementByName("fromPort"));

Select destination = new Select(driver.findElementByName("toPort"));

departure.selectByVisibleText("Boston");

destination.selectByVisibleText("London");

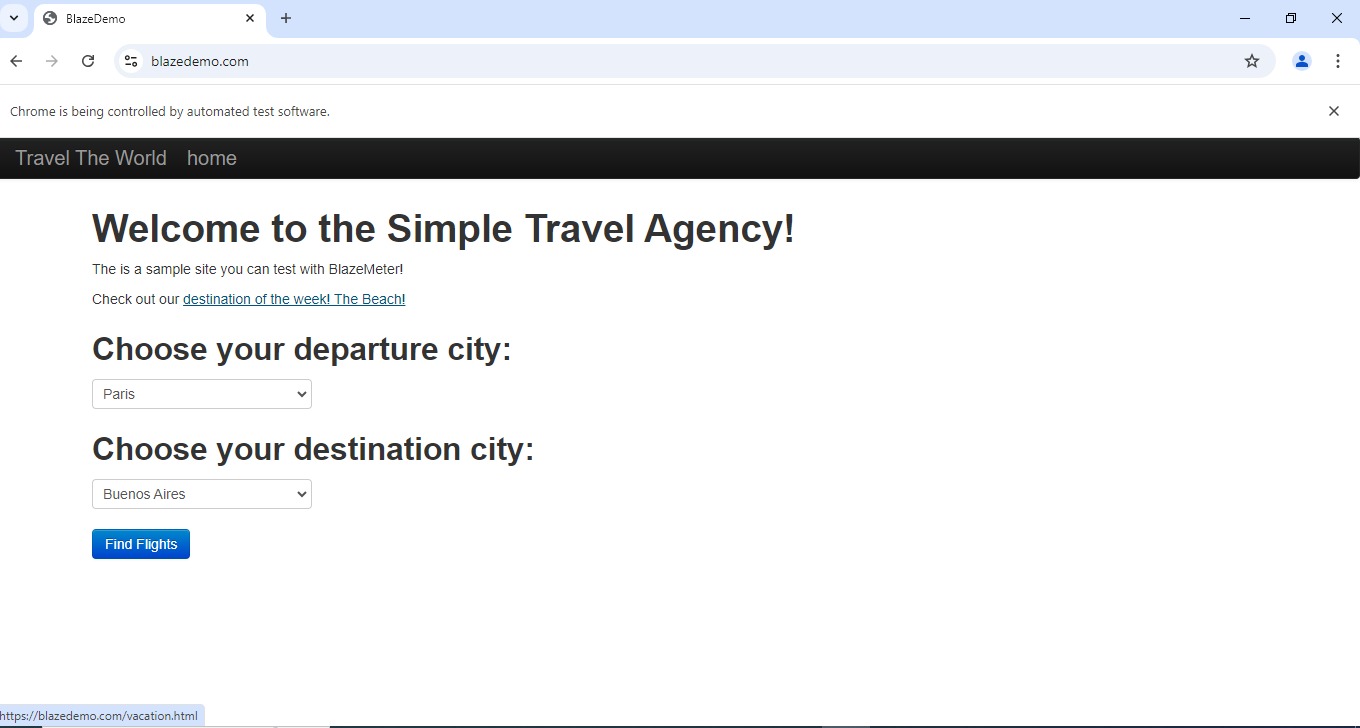
Thread.sleep(2000);

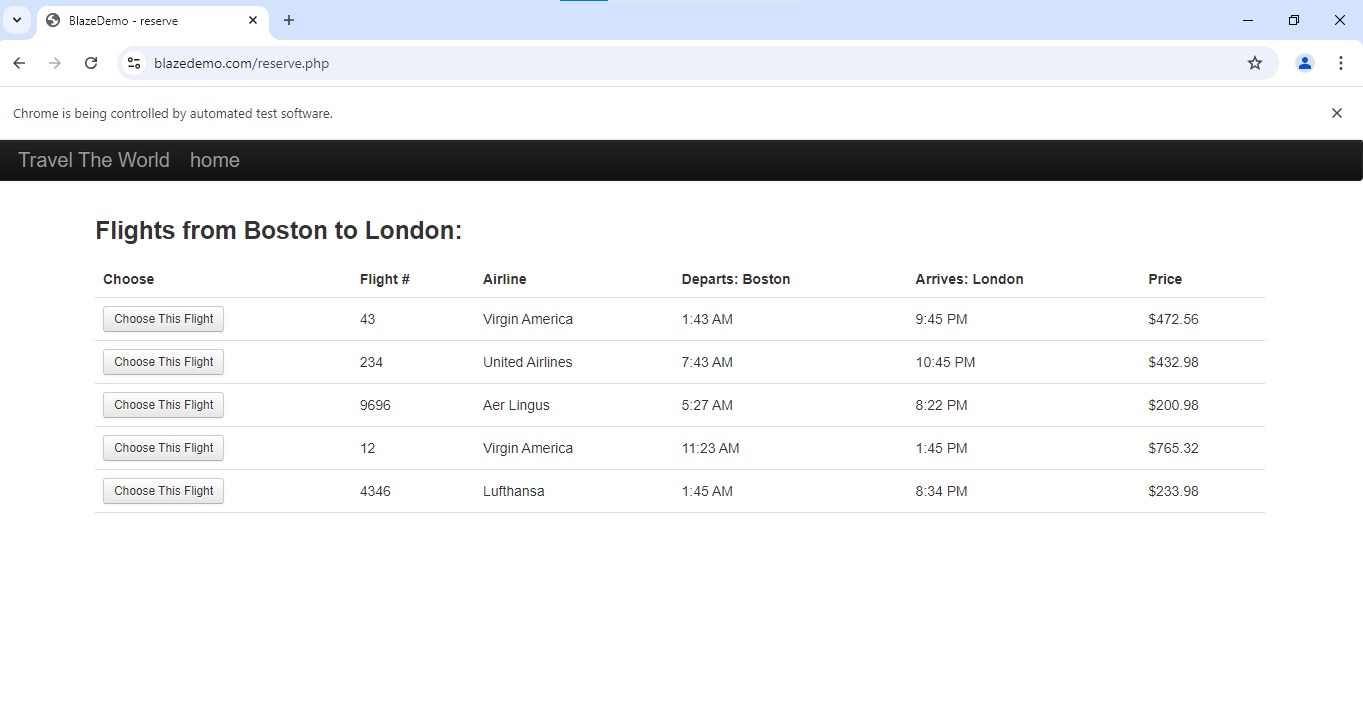
driver.findElementByCssSelector("body > div.container > form > div > input").click();

Thread.sleep(2000);

driver.close(); }}

OUTPUT:-





Practical 7

Aim: Demonstrate CheckBox and Radio Button in Selenium WebDriver and Testng.

CODE:-

package demoselenium;

//import org.testng.annotations.Test;

//import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebDriver.Options;

import org.openqa.selenium.WebDriver.Window;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class testdemo {

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

System.*setProperty*("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

Options manage = driver.manage();

Window = manage.window();

window.maximize();

driver.get("https://www.htmldog.com/examples/inputcheckboxes.html");

WebElement drama = driver.findElementByName("drama");

WebElement action = driver.findElementByName("action");

WebElement comedy = driver.findElementByName("comedy");

WebElement horror = driver.findElementByName("horror");

WebElement scifi = driver.findElementByName("scifi");

comedy.click();

scifi.click();

WebElement lt20 = driver.findElementById("lt20");

WebElement gt20tolt40 = driver.findElementById("20to39");

WebElement gt40tolt59 = driver.findElementById("40to59");

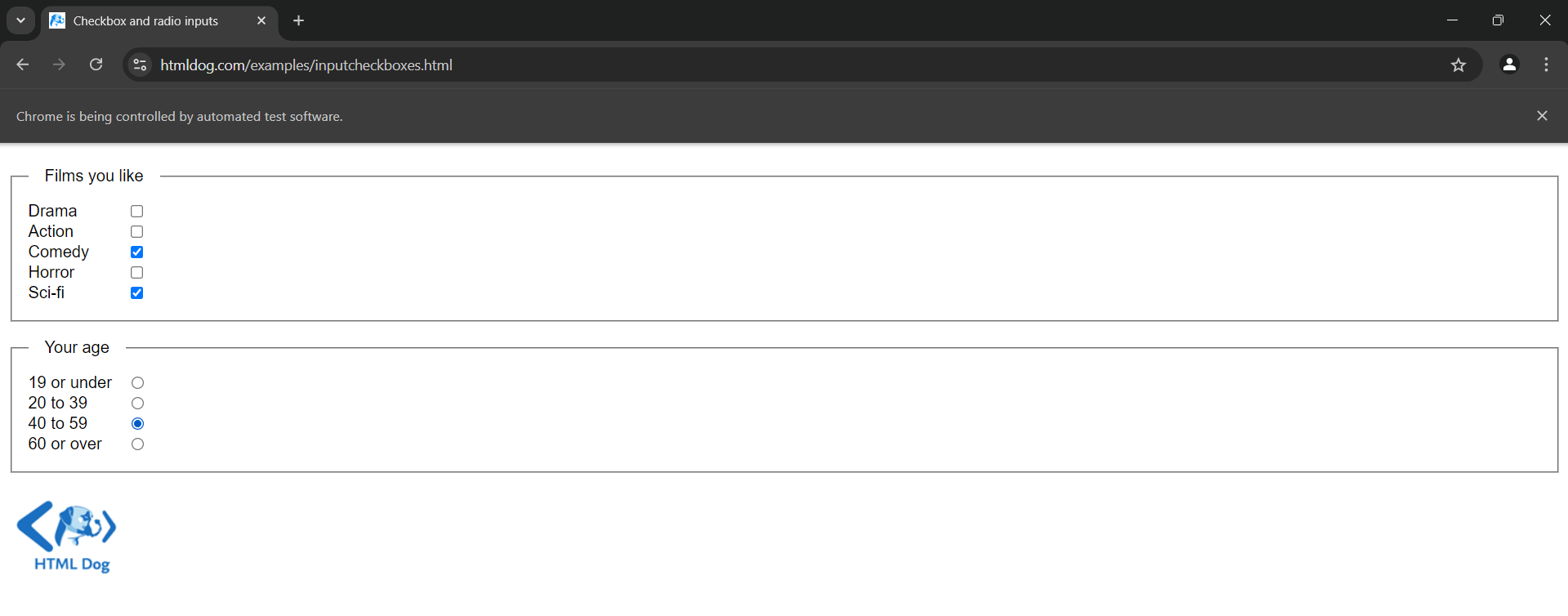
WebElement gt59 = driver.findElementById("gt59");

gt40tolt59.click();

Thread.*sleep*(3000);

driver.close(); }}

OUTPUT:-



Practical 8

Aim: Implementing Selenium WebDriver – find element command, Locator(Class name, css selector, Xpath)

CODE:-

package demoselenium;

//import org.testng.annotations.Test;

//import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebDriver.Options;

import org.openqa.selenium.WebDriver.Window;

//import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class testdemo {

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

System.*setProperty*("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

Options manage = driver.manage();

Window = manage.window();

window.maximize();

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

//driver.findElementByClassName("gLFyf").sendKeys("Mumbai University");

//driver.findElementByCssSelector("#APjFqb").sendKeys("Mumbai University");

driver.findElementByXPath("//\*[@id=\"APjFqb\"]").sendKeys("University of Mumbai");

driver.findElementByCssSelector("body > div.L3eUgb > div.o3j99.ikrT4e.om7nvf > form > div:nth-child(1) > div.A8SBwf > div.FPdoLc.lJ9FBc > center > input.gNO89b").click();

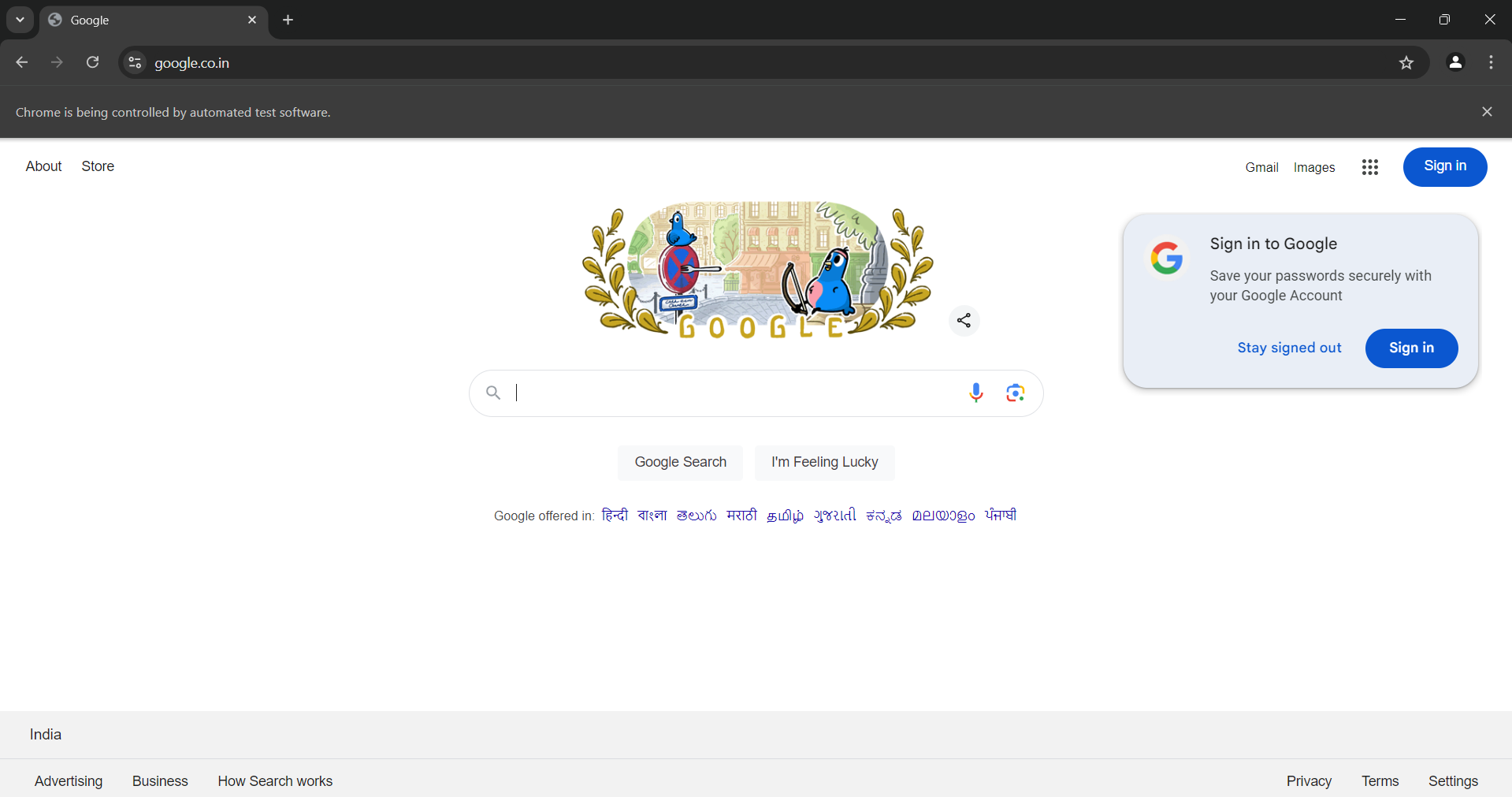
driver.findElementByXPath("//\*[@id=\"rso\"]/div[1]/div/div/div/div/div/div/div/div[1]/div/span/a").click();

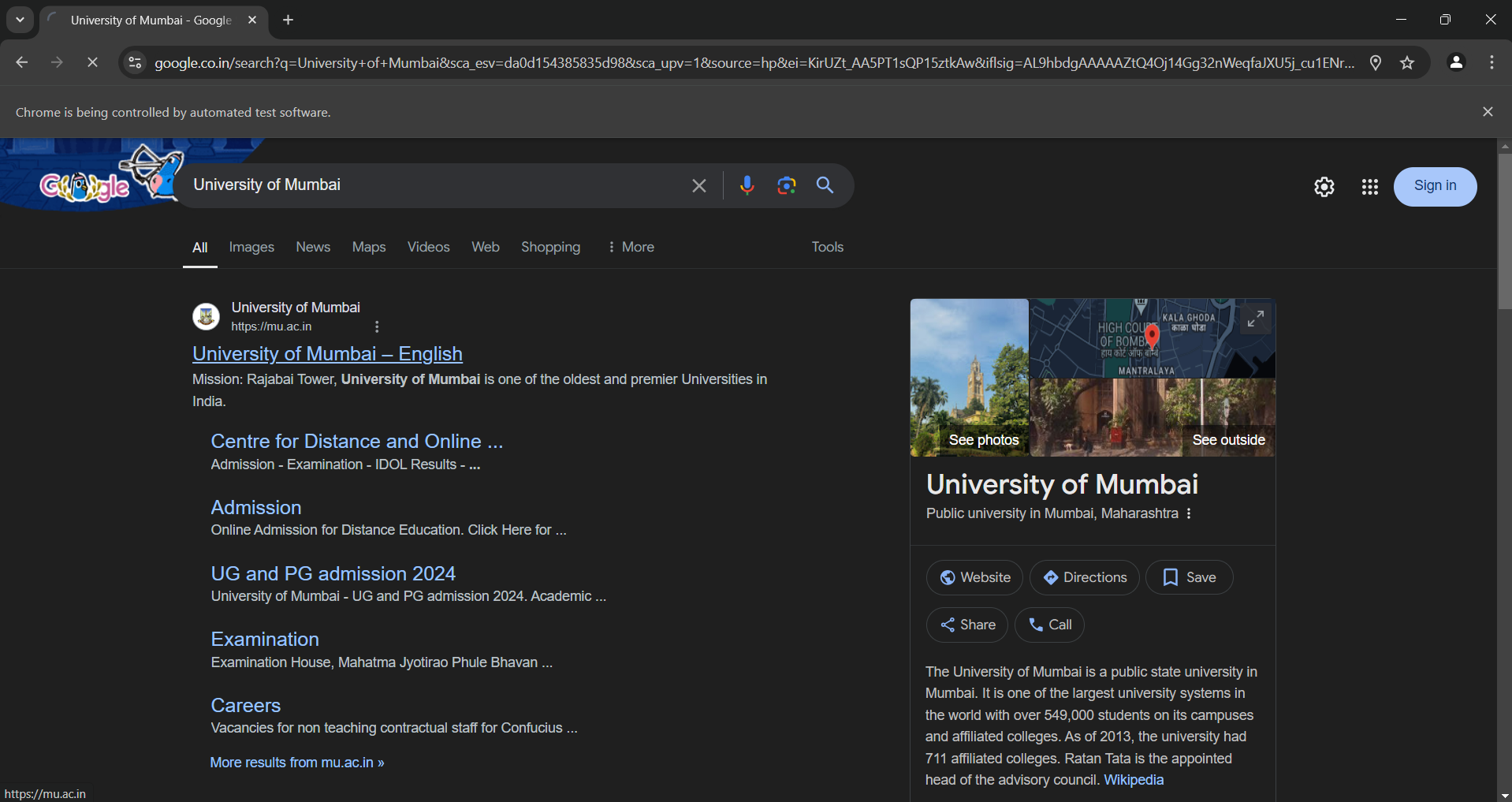
Thread.*sleep*(2000);

driver.close();}

}

OUTPUT:-







Practical 9

Aim: Demonstrate login form in Selenium WebDriver and Testng.

CODE:-

package demoselenium;

//import org.testng.annotations.Test;

//import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebDriver.Options;

import org.openqa.selenium.WebDriver.Window;

//import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

public class testdemo {

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

System.*setProperty*("webdriver.chrome.driver", "./drivers/chromedriver.exe");

ChromeDriver driver = new ChromeDriver();

Options manage = driver.manage();

Window = manage.window();

window.maximize();

driver.get("https://practicetestautomation.com/practice-test-login/");

driver.findElementByName("username").sendKeys("student");

driver.findElementByName("password").sendKeys("Password123");

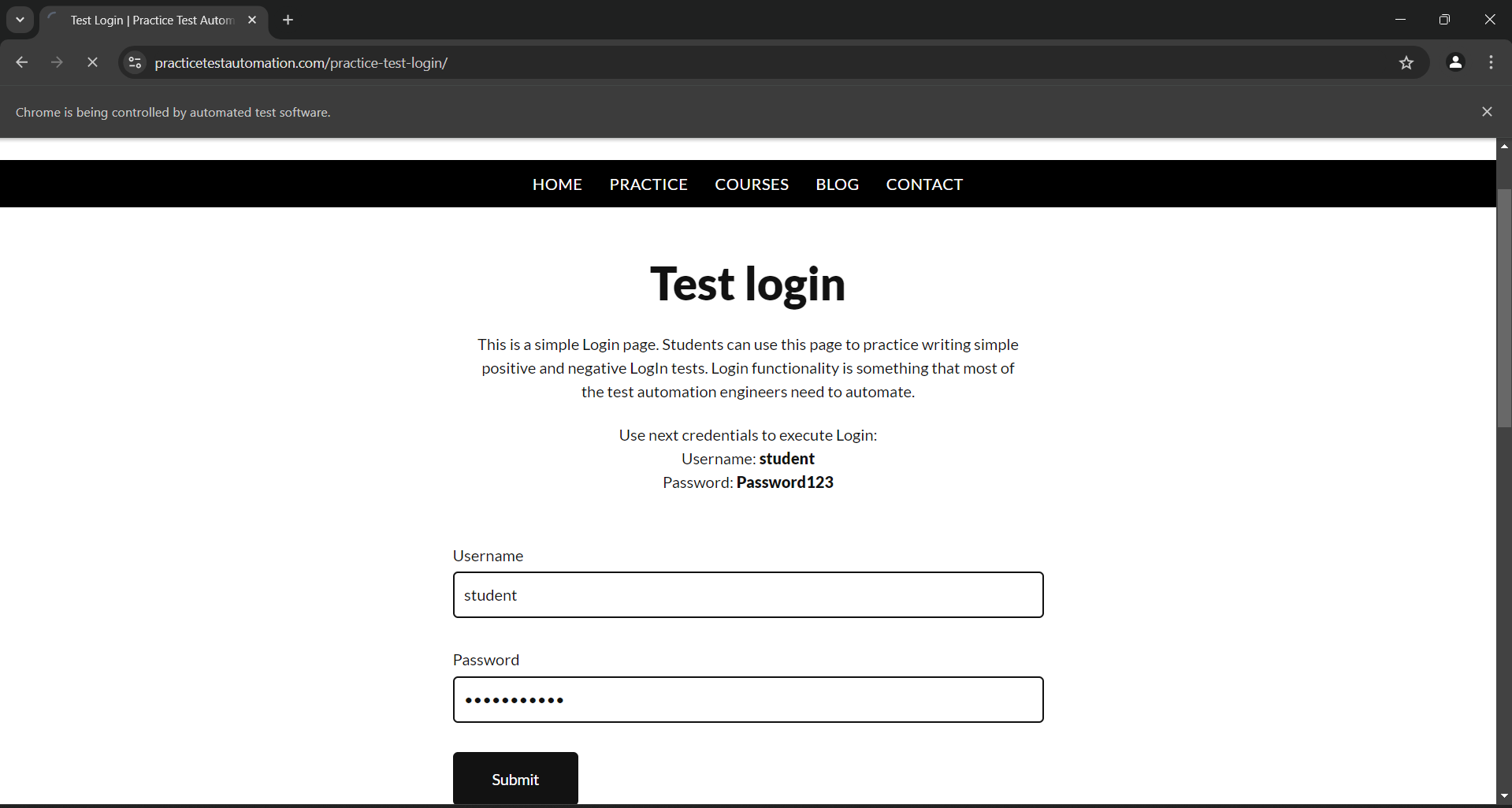
driver.findElementById("submit").click();

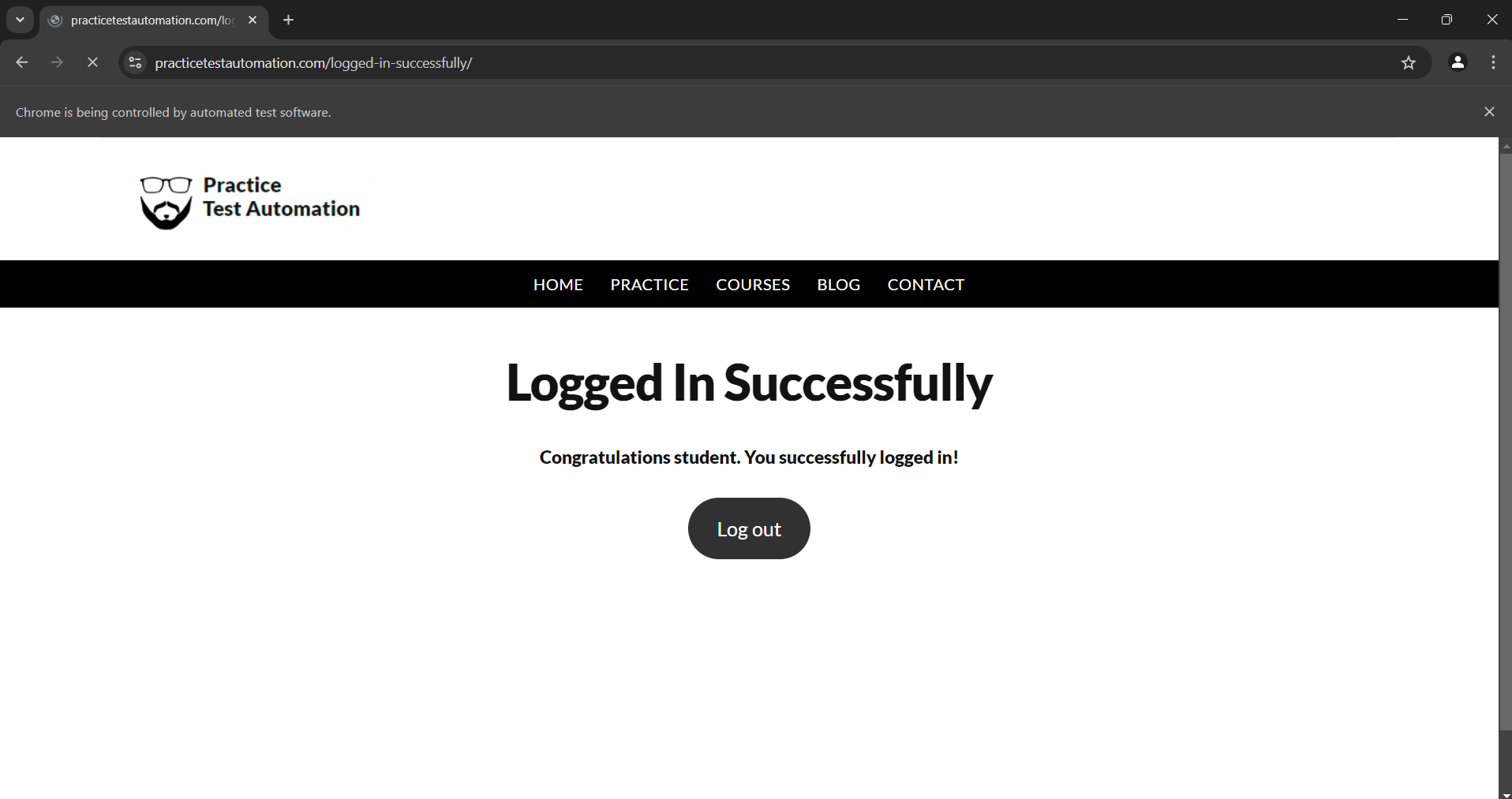
Thread.*sleep*(2000);

driver.close();

}}

OUTPUT:-





Practical 10

Aim: Preparing a Report on test case for given scenario

A]

Aim: Functional Testing using Boundary Value Analysis

Scenario: A program takes the value of age from 21 to 65. Design test cases using boundary value analysis.

|  |  |  |
| --- | --- | --- |
| **BOUNDARY VALUE TEST CASE** | | |
| INVALID TEST CASE  (Min Value - 1) | VALID TEST CASE  (Min, +Min, Max, -Max) | INVALID TEST CASE  (Max Value + 1) |
| 20 | 21, 22, 65, 64 | 66 |

**Test Case Scenario**

1. Input: - Enter the value of age as 20(21 - 1)

Output: - Invalid

1. Input: - Enter the value of age as 21

Output: - Valid

1. Input: - Enter the value of age as 22(21 + 1)

Output: - Valid

1. Input: - Enter the value of age as 65

Output: - Valid

1. Input: - Enter the value of age as 64(65 - 1)

Output: - Valid

1. Input: - Enter the value of age as 66(65 + 1)

Output: - Invalid

B]

Aim: Functional Testing using Equivalence Partitioning

Scenario: 3% rate of interest is given if the balance in the account is in the range of $0 to $100, 5% rate of interest is given if the balance in the account is in the range of $100 to $1000, and 7% rate of interest is given if the balance in the account is $1000 and above, we would initially identify three valid equivalence partitions and one invalid partition as shown below.

**Partition 1:** balance 0-100

Valid Inputs - 0-100- >=0 And <=100

Invalid Input-<0, $,#,@ A-Z

**Partition 2**: balance 100-1000

Valid Input - 100-1000 (>100 And <=1000)

Invalid Input- $,#,@ A-Z

**Partition 3**: balance >1000

Valid Input- >1000

Invalid Input- $,#,@ A-Z

Invalid Valid partitions Valid Partition

<0 0-100 100-1000 >1000

3% Interest 5% Interest 7% Interest

**Test Case**

**Test case id Test input Expected**

Account Balance Output

B001 -90 Invalid input

B002 50 3% Interest

B003 900 5% Interest

B004 2500 7% Interest

B005 A Invalid input

B006 $ Invalid input

C]

Aim: Functional Testing using Decision Table

Consider test cases based on decision table for a ‗Login‘ Page Functionality.

Business Rules:

1. On entering correct combination of ID & Password, user should be able to login successfully.

2. User is not allowed to login when any or both of the ID & Password are incorrect /blank. In such cases, it should show ‗Invalid

Credentials‘ message.

**We created the following combinations of Conditions, Actions and the respective rules in the decision table.**

****

In the above table, there are

1. conditions – UserID, Password

2. Actions – Login Successfully, Error showing ‗Invalid Credentials‘ and

3. Options — Blank, Valid, Invalid.

So, the total number of test cases are as follows:

**Options conditions i.e 32 = 9 Test cases**

All test cases are not valid and significant some we need to optimise the test cases

Rules 1, 2, 3, 4, and 5 cover the same action Item ―Invalid Credentials‖ with options Blank and Invalid. Hence, we can consider any one of these test cases TC01 OR TC02 OR TC03 OR TC04 OR TC05

1. Rules 6,7, and 8 cover the same action Item ―Invalid Credentials‖ with options Valid and Invalid. Hence, we can consider any of these test cases TC06 OR TC07 OR TC08

2. Rule 9 covers the action item ―Login Successfully‖ with all valid options. Hence, we should consider the test case TC09.

**Condensed Decision Table as shown below:******

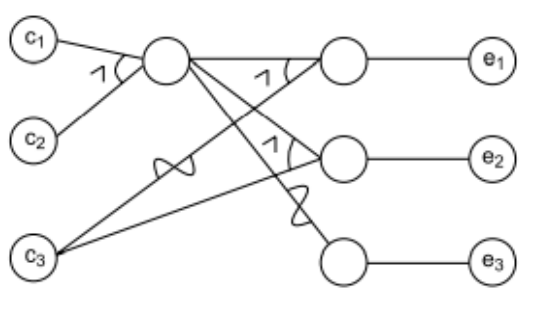
D]

Aim: Functional Testing using Cause Effect Graph

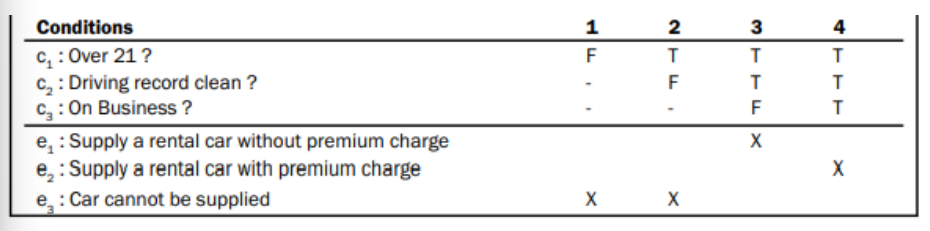
A tourist of age greater than 21 years and having a clean driving record is supplied a rental car. A premium amount is also charged if the tourist is on business, otherwise it is not charged. If the tourist is less than 21 year old, or does not have a clean driving record, the system will display the following message: ―Car cannot be supplied‖ Draw the cause-effect graph and generate test cases

Solution: The causes are c1 : Age is over 21 c2 : Driving record is clean c3 : Tourist is on business and effects are e1 : Supply a rental car without premium charge. e2 : Supply a rental car with premium charge e3 : Car cannot be supplied .The Cause Effect Graph and test cases based on that are as follows :

Cause Effect Graph:



Decision Table on Rental Car Problem:



Test Cases:

