Roll No.**24** Exam Seat No._____

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

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

Since 1962

CERTIFICATE

Certified that Mr. **NARENDER KESWANI [ROLL NO: 24]** of **SYMCA-1B** has satisfactorily completed a course of the necessary experiments in **MCAL35 – Software Testing Quality Assurance Lab** under the supervision of **Dr. Meeanakshi Garg** in the Institute of Technology in the academic year **2022- 2023.**

Principal Head of Department

Lab In-charge Subject Teacher

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

Department of M.C.A

MCAL35 – SOFTWARE TESTING QUALITY ASSURANCE LAB INDEX

Sr. No	Contents	Date Of Preparation	Date Of Submission	Marks	Sign
1	Take a review and write test cases for any known application.	25/08/2022	29/08/2022	10	
2	Implement Web Drivers on Chrome & Firefox Browsers.	29/08/2022	08/09/2022	10	
3	Demonstrate handling multiple frames in selenium.	08/09/2022	15/09/2022	10	
4	Implement Browser command and navigation Commands.	15/09/2022	22/09/2022	10	
5	Implement the find element command.	29/09/2022	06/10/2022	10	
6	Demonstrate the Locator. (id, css selector, path)	29/09/2022	06/10/2022	10	
7	Demonstrate synchronization in selenium.	29/09/2022	06/10/2022	10	
8	Demonstrate different types of alerts.	29/09/2022	06/10/2022	10	
9	Demonstrate: Handling Drop Down & List Boxes.	06/10/2022	13/10/2022	10	
10	Demonstrate Command Button Radio buttons & text boxes.	13/10/2022	20/10/2022	10	
11	Demonstrate action classes in Selenium.	13/10/2022	20/10/2022	10	
12	Installation of TestNg, running TestNg and TestNg annotations.	20/10/2022	27/10/2022	10	
13	Demonstrate data driven Framework.	20/10/2022	27/10/2022	10	
A1	Assignment I: Software Basics - I	29/09/2022	06/10/2022	10	
A2	Assignment II: Software Basics – II	29/09/2022	06/10/2022	10	
A3	Assignment III: Software Basics - III	29/09/2022	06/10/2022	10	

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

Project:

E-Learning & Learning Management System (LMS)

TechStack:

HTML, CSS, BOOTSTRAP, JAVASCRIPT, PHP, etc

Tools Used:

Wordpress, Elementor, etc

Project Description:

A learning management systems (LMS) is a software application that reports, tracks and tests learners so that learners can progress through online courses. The learning management system emerged from e-Learning.
It is a fully-managed eLearning solution including a student interface, course modules for students, zoom classes and email notifications. It provides a dynamic and customizable platform to meet specific educational needs for our customers.
It allows tutors to manage their online education and virtual classrooms. It can be used on any screen and is built to provide the same experience on any screen size, whether that be on a mobile, tablet or PC screen. Tutors can create quizzes for students, and they can build these smart quizzes for exams that can be autocorrected and graded with feedback.

Major Functionalities:

- Lessons, Quiz & Assignments: Create compelling quizzes, assignments, lessons, and more to create resourceful eLearning courses.
- Content Drip: Get full control over learning path management with powerful included addons like Course Content Drip.
- Personalized Dashboard for Teachers & Students: Organized and personalized dashboard for teachers & students. Access everything you need to manage your LMS website from one spot.
- Unlimited Quizzes: Powerful quiz creator to create custom-tailored quizzes with handy features like unique question types, timer, and more to create compelling quizzes.

- Assignments: Improve learners' retention with easy to create and assess assignments features for your courses without any hassle and make eLearning more effective & engaging.
- Event Calender: Let students easily keep track of assignments. Schedule assignments set deadlines and keep students up to date with your eLearning plan.
- Analytics & Reports: Assess the effectiveness of the courses, track the performance of individual student progress, and more with data-driven insights.

Test Case 1: Verify login with valid email and password of maarula website

Description: Test the Maarula Classes login page

Pre-Conditions: User has valid email and password

URL: https://maarula.in/dashboard/

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to login page		User should be able to login	User is navigated to login page	Pass
2	Provide valid email address	xebixik967@esmoud.com	Should accept input data	Input accepted	Pass
		xebixik967esmoud.com	Enter valid email address	Enter valid email address	Fail
3	Provide valid password	Narender@123	Should accept input data	Input accepted	Pass
		narender123	Not a strong password	Not a strong password	Fail
4	Click on Login button		Signed In to User Profile & navigate to Home Page	Signed In to User Profile & navigate to Home Page	Pass

Post-Conditions: User is validated with the database and successfully logs into the account. The account session details are logged in the database.

Test Case 2: Verify the entered details of registration page of maarula website

Description: Test the Maarula Classes Registration page

Pre-Conditions: User should have valid registration details.

URL: https://maarula.in/student-registration

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to registration page		User is navigated to registration page	User is navigated to registration page	Pass
2	Provide valid first name	"Narender"	Should accept the input data	Input Accepted	Pass
		"user20@dcw %^"	Special Characters and numbers in first name are not allowed	Special Characters and numbers in first name are not allowed	Fail
3	Provide valid last name	"Keswani"	Should accept the input data	Input Accepted	Pass
		"dragon10@ &*()dw"	Special Characters and numbers in last name are not allowed	Special Characters and numbers in last name are not allowed	Fail
4	Provide Valid User Name	"narender"	Should accept the input data	Input Accepted	Pass

		"@#sdvvc"	Special Characters in username is not allowed	Special Characters in username is not allowed	Fail
5	Provide Valid E-mail	"narender.rk10@gmail.com"	Should accept the input data	Input Accepted	Pass
		"narender.rk10gmail.com"	Invalid Email Address	Invalid Email Address	Fail
6	Provide Valid Password	"Abc89@25"	Should accept the input data	Input Accepted	Pass
		"abc8925"	Password should contains at least one uppercase character, one lowercase character , one numerical value , one special character and length should be greater or equal to the 8	Password should contains at least one uppercase character, one lowercase character , one numerical value , one special character and length should be greater or equal to the 8	Fail
7	Register Button		User should be registered	User should be registered	Pass

Post-Conditions:: User is validated with the data and successfully registered into the account. The account details are stored in the database.

Test Case 3: Verify the entered details on contact page of maarula website

Description: Test the Maarula Classes Contact Us page

Pre-Conditions: User should have valid input details.

URL: https://maarula.in/contact-us/

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to contact us page		User should be able to fill the contact form	User is navigated to contact us page	Pass
2	Provide Full Name	"Narender Keswani"	Should accept input data	Input accepted	Pass
		""	This field is required	This field is required	Fail
3	Provide valid email address	xebixik967@esmoud.com	Should accept input data	Input accepted	Pass
		xebixik967esmoud.com	Enter valid username	Enter valid username	Fail
4	Provide valid mobile number	9320907041	Should accept input data	Input accepted	Pass
		5145415454584	Mobile number can't be more than 10 characters	Mobile number can't be more than 10 characters	Fail
5	Provide Subject	"Enquiry"	Should accept input data	Input accepted	Pass
		""	This field is required	This field is required	Fail
6	Provide Message	"I require enquiry about this course"	Should accept input data	Input accepted	Pass

		“”	This field is required	This field is required	Fail
7	Click on Submit button		User Query will be recorded	User Query will be recorded	Pass

Post-Conditions:: User is validated with the data and data will be stored into the account and the email will be sent to the admin.

Test Case 4: Verify the entered details on forgot password of maarula website

Description: Test the Maarula Classes Forgot Password page

Pre-Conditions: User should have valid input details.

URL: **https://maarula.in/dashboard/retrieve-password/**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to forgot password page		User should be able to navigate	User is navigated to forgot password page	Pass
2	Provide valid email address	xebixik967@esmoud.com	Should accept input data	Input accepted	Pass
		xebixik967esmoud.com	Enter valid email address	Enter valid email address	Fail
3	Click on Reset Password button		System will send the reset link to the user	System will send the reset link to the user	Pass

Post-Conditions:: User is validated with the data and if the email is matched with database records then a password reset link will be sent to the user.

Test Case 5: Verify the entered details on change password of maarula website

Description: Test the Maarula Classes Create Course page

Pre-Conditions: User should have valid input details.

URL: https://maarula.in/dashboard/settings/reset-password/

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to change password page		User should be able to navigate change password page	User is navigated to change password page	Pass
2	Provide valid old password	"********"	Should accept input data	Input accepted	Pass
		""	Old Password can't be empty	Old Password can't be empty	Fail
3	Provide valid new password	"Abc89@25"	Should accept the input data	Input Accepted	Pass
		"abc8925"	Password should contains at least one uppercase character, one lowercase character , one numerical value , one special character and length should be greater or equal to the 8	Password should contains at least one uppercase character, one lowercase character , one numerical value , one special character and length should be greater or equal to the 8	Fail

4	Click on Change Password button		System will change password of the user	System will change password of the user	Pass

Post-Conditions: If the old password of the user is matched with the database then the new password will be updated in the database.

Test Case 6: Testing of creating course by admin of maarula website

Description: Test the Maarula Classes Create Course page

Pre-Conditions: Admin should have valid input details.

URL: https://maarula.in/wp-admin/post-new.php?post_type=courses

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to create course page		Admin should be able to navigate to course page	Admin is navigated to create course page	Pass
2	Provide valid course details	Maximum Students, Enrollment Expiration, Difficulty Level, Course Type, Total Course Duration, Benefits of the course, Course Intro Video	Should accept input data	Input accepted	Pass
3	Provide valid topics details of course	Topics and sub topics of the course	Should accept input data	Input accepted	Pass
4	Click on Create Course button		System will create course	System will create course	Pass

Post-Conditions: Admin will create the course and details of the course will be stored in the database.

Test Case 7: User searches the course on maarula website

Description: Test the Maarula Classes Courses page

Pre-Conditions: User should have valid input details.

URL: **https://maarula.in/courses/**

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to course page		User should be able to navigate to courses page	User is navigated to courses page	Pass
2	User should able to see all the courses		Courses will be displayed on the web page	Courses will be displayed on the web page	Pass
3	Provide course name for search	"MCA 2023"	Should accept input data	Input accepted	Pass
		""	This field can't be empty	This field can't be empty	Fail
4	Click on Search Course button		System will display results	System will display results	Pass

Post-Conditions: Users will get the course details based on their search query.

Test Case 8: User registers the course on maarula website

Description: Test the Maarula Classes Course Register page

Pre-Conditions: User should have valid input details.

URL: https://maarula.in/courses/

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)
1	Navigate to course page		User should be able to navigate to courses page	Admin is navigated to courses page	Pass
2	User should able to see all the courses		Courses will be displayed on the web page	Courses will be displayed on the web page	Pass
3	User select one course for purchase	Course_id = "mca-2023" User_id = "123"	Selected course will be added to the cart	Selected course will be added to the cart	Pass
4	Click on Buy Course button		System will redirect to payment gateway	System will redirect to payment gateway	Pass
5	Provide valid card, bank transfer details	Card number: 1000 2000 3000 4000 Name: Narender Keswani Expire Date: 25/12/2023 CVV: 123	Should accept input data	Input accepted	Pass
	Provide invalid card, bank transfer details	Card number: 1000 2000 3000 Name: Narender Expire Date: 25/13 CVV: 120	Enter valid details	Enter valid details	Fail

6	User will confirm purchase		Course will be purchased and details will be notified to the user	Course will be purchased and details will be notified to the user	Pass

Post-Conditions: Users will get access to the course after purchasing the course.

Conclusion:

The usability evaluation of website design implemented successfully.

Aim: Implement Web Drivers on Chrome & Firefox Browsers.

THEORY:

WebDriver

WebDriver drives a browser natively, as a user would, either locally or on a remote machine using the Selenium server, marks a leap forward in terms of browser automation.

Selenium WebDriver refers to both the language bindings and the implementations of the individual browser controlling code. This is commonly referred to as just WebDriver.

Selenium WebDriver is a W3C Recommendation

- WebDriver is designed as a simple and more concise programming interface.
- WebDriver is a compact object-oriented API.
- It drives the browser effectively.

DOWNLOAD URL OF DRIVERS / LIBS:

- SELENIUM DRIVER: https://github.com/SeleniumHQ/selenium/releases/download/selenium-4.4.0/selenium-java-4.4.0.zip
- CHROME DRIVER: https://chromedriver.storage.googleapis.com/105.0.5195.52/chromedriver_win32.zip
- GECKODRIVER: https://github.com/mozilla/geckodriver/releases

SOURCE CODE:

```
package stqaprac2;

/*
 * @author NARENDER KESWANI
 */

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

public class STQAPrac2 {

static WebDriver wd;
public static void main(String[] args) {
System.setProperty("webdriver.chrome.driver","C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
Scanner sc = new Scanner(System.in);
```

```
System.out.println("1.ChromeBrowser \n2. Firefox Broswer");
System.out.println("Choice");
int ch = sc.nextInt();
sc.close();
switch (ch)
{
case 1:
System.setProperty("webdriver.chrome.driver","C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
wd = new ChromeDriver();
break;
case 2:
System.setProperty("webdriver.gecko.driver","C:\\Users\\NARENDER
KESWANI\\Downloads\\geckodriver-v0.31.0-win64\\geckodriver.exe");
wd = new FirefoxDriver();
default:
System.out.println("Invalid Choice");
break;
}
if(wd!=null)
{
wd.get("http:\\google.com");
}
}
}
```

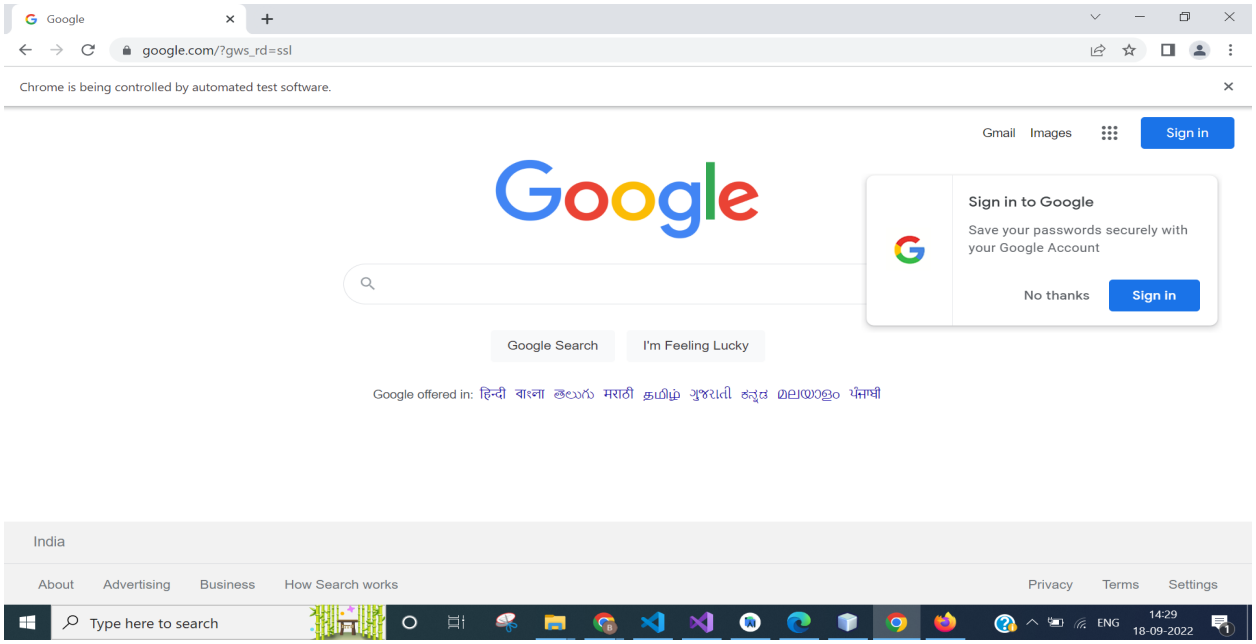
OUTPUT:

```
run:
1.ChromeBrowser
2. Firefox Broswer
Choice
3
Invalid Choice
BUILD SUCCESSFUL (total time: 1 second)
```

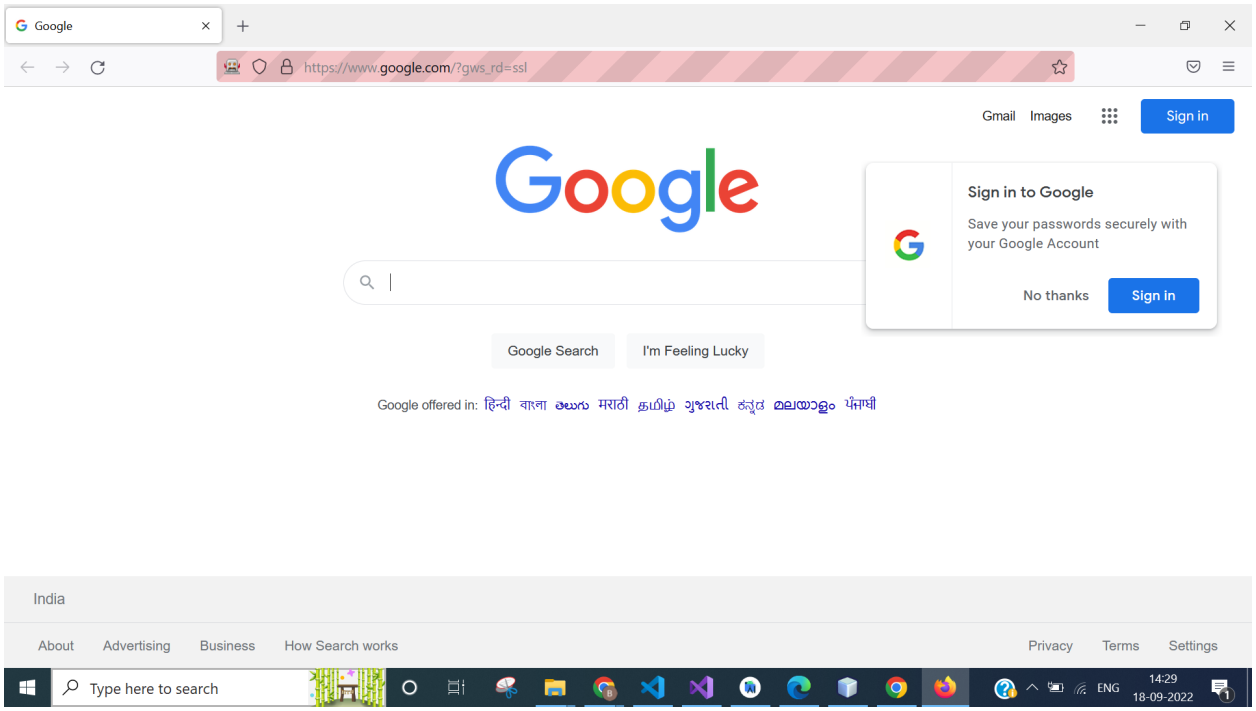
INVALID CHOICE:

```
run:
1.ChromeBrowser
2. Firefox Broswer
Choice
3
Invalid Choice
BUILD SUCCESSFUL (total time: 4 seconds)
```

CHOICE 1 : GOOGLE CHROME



CHOICE 2: FIREFOX



CONCLUSION:

Web Drivers on Chrome & Firefox Browsers are implemented.

Aim: Demonstrate handling multiple frames in selenium

Theory:

iFrame in Selenium Webdriver

iFrame in Selenium Webdriver is a web page or an inline frame which is embedded in another web page or an HTML document embedded inside another HTML document. The iframe is often used to add content from other sources like an advertisement into a web page. The iframe is defined with the <iframe> tag.
We can identify the frames in Selenium using methods given below:

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

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

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

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

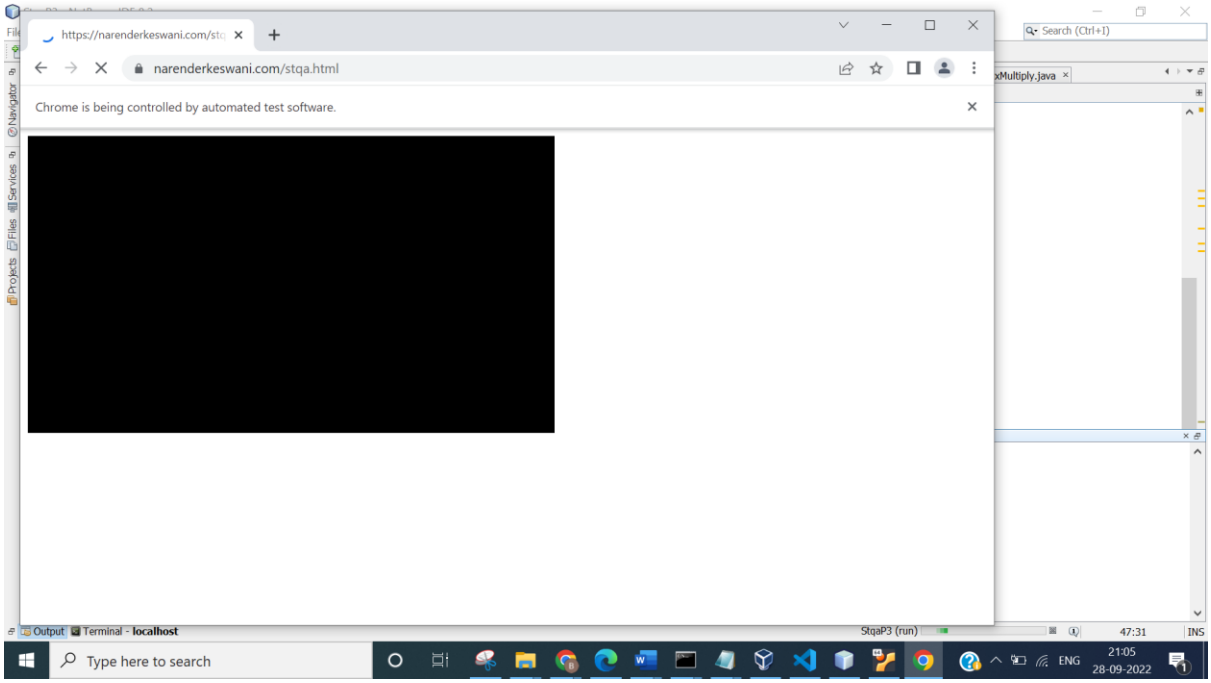
SOURCE CODE:

```
/**
 *
 * @author NARENDER KESWANI
 */
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

public class P3 {

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

                System.setProperty("webdriver.chrome.driver","C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");

                driver = new ChromeDriver();
                driver.get("https://narenderkeswani.com/stqa.html");
```

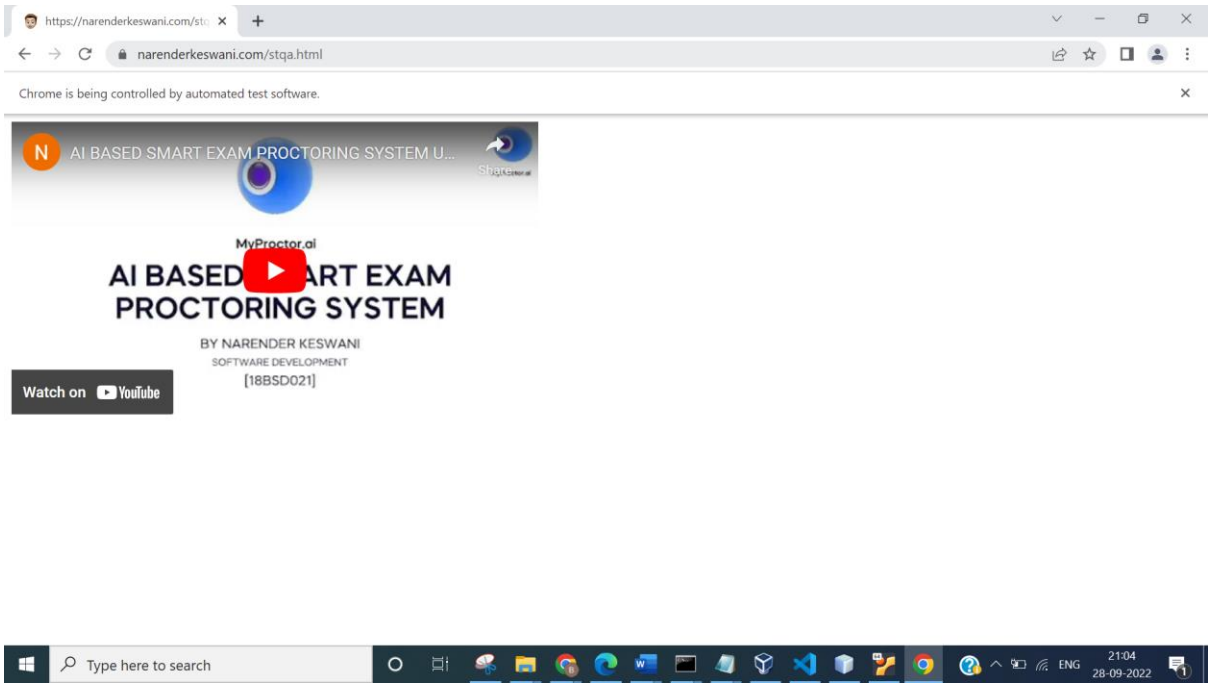
```
                // navigates to the page consisting an iframe
                driver.manage().window().maximize();
                Thread.sleep(5000);

                System.out.println("Switching to iframe");
        driver.switchTo().frame("stqa");
        Thread.sleep(5000);

        System.out.println("Switched iframe sucessfully!");

        System.out.println("Playing video!");
                driver.findElement(By.xpath("/html/body/div/div/div[4]/button")).click();
                Thread.sleep(2000);

        System.out.println("Sharing video!");
        driver.findElement(By.xpath("/html/body/div[1]/div/div[3]/div[3]/button[2]")).click();
                Thread.sleep(5000);

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

OUTPUT:

Loading website:

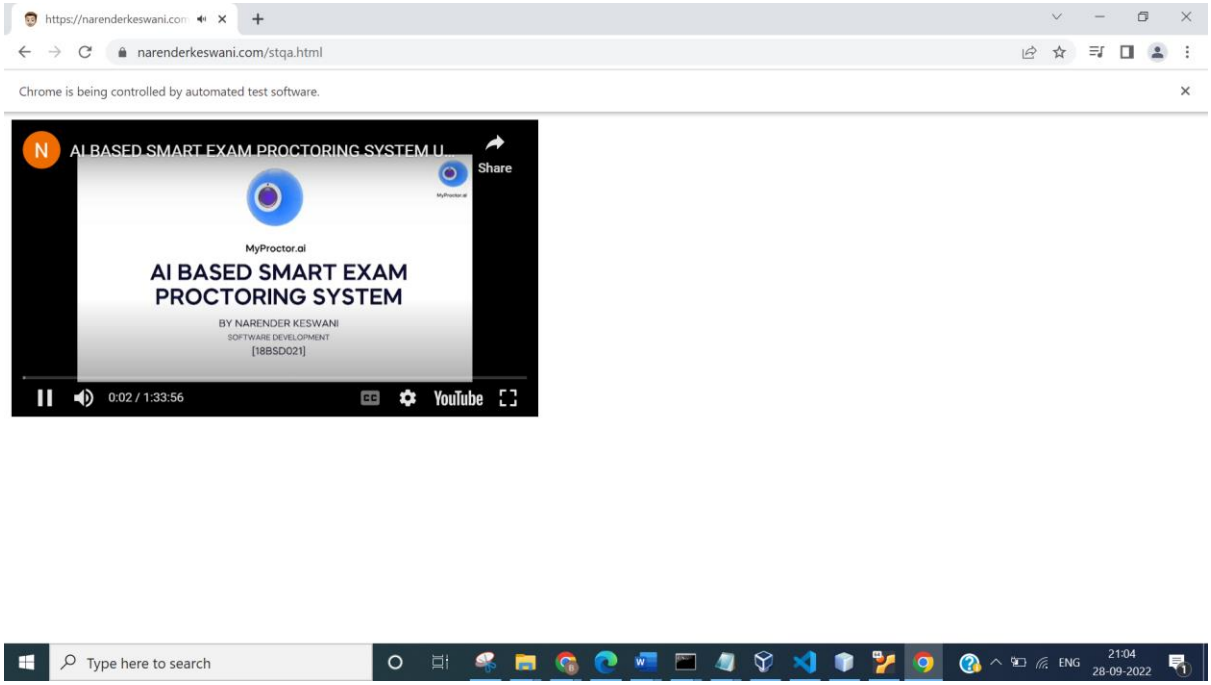


Maximizing page:

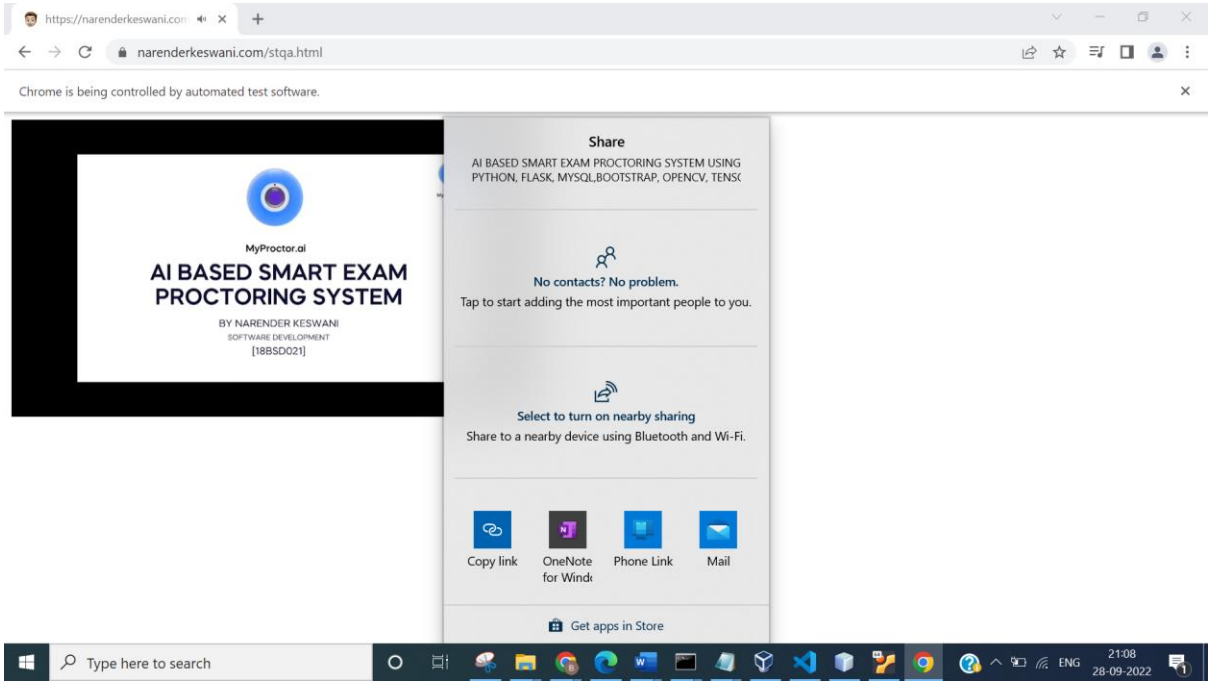


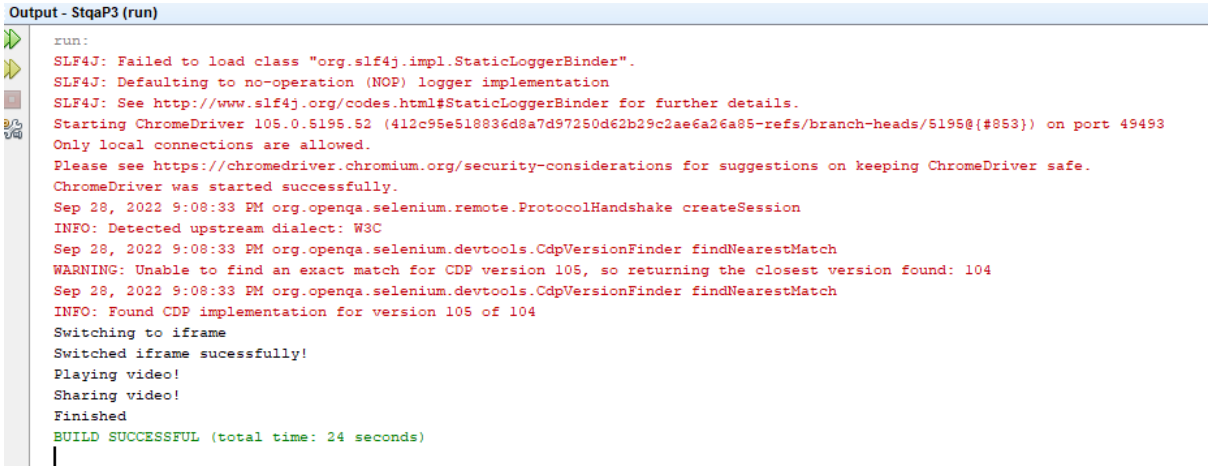
SWITCHING TO IFRAME

PLAYING VIDEO



SHARING VIDEO:





CONCLUSION:

Demonstration of multiple frames has been implemented using selenium.

Aim: Implement Browser command and navigation Commands.

Theory:

Selenium WebDriver - Browser Commands:

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

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

1. **Get Command:**
   Method: get(String arg0) : void
   In WebDriver, this method loads a new web page in the existing browser window. It accepts String as parameter and returns void.

   Implementation:

```
driver.get(URL);
OR
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.

   Implementation:

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

3. **Get Current URL Command:**
   Method: getCurrentUrl(): String
   In WebDriver, this method fetches the string representing the Current URL of the current web page. It accepts nothing as parameter and returns a String value.

   Implementation:

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

4. **Close Command:**
   Method: close(): void

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

Implementation:

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

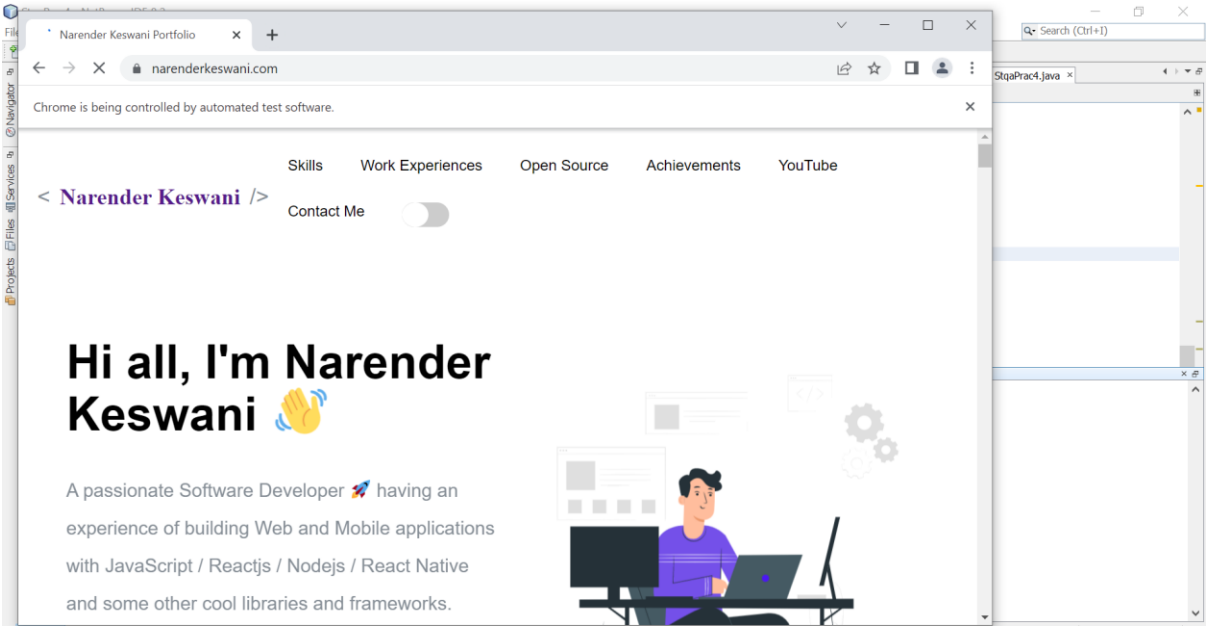
SOURCE CODE:

```
package stqaprac4;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
/**
 * @author NARENDER KESWANI
 */
public class StqaPrac4 {

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

      System.setProperty("webdriver.chrome.driver","C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
      driver = new ChromeDriver();

      String appUrl = "https://narenderkeswani.com/";
      driver.get(appUrl);
      Thread.sleep(3000);

      // Click on Contact Us
      driver.findElement(By.xpath("/html/body/div/div/div/div[1]/div/header/ul/li[6]/a")).click();
      Thread.sleep(3000);

      // Go back to Home Page
      driver.navigate().back();
      Thread.sleep(3000);

      // Go forward to  Contact Us
      driver.navigate().forward();
      Thread.sleep(3000);

      // Go back to Home page
      driver.navigate().to(appUrl);
      Thread.sleep(3000);

      // Refresh browser
      driver.navigate().refresh();
```

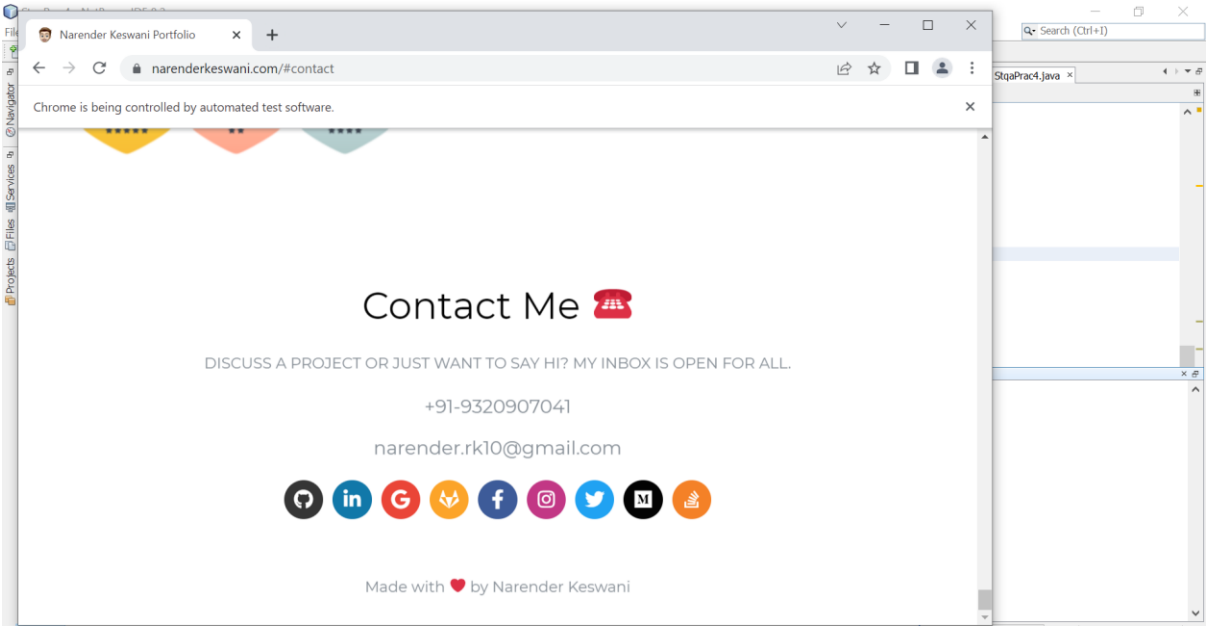
```
        Thread.sleep(3000);
        // Close browser
        driver.close();
    }
}
```

OUTPUT:

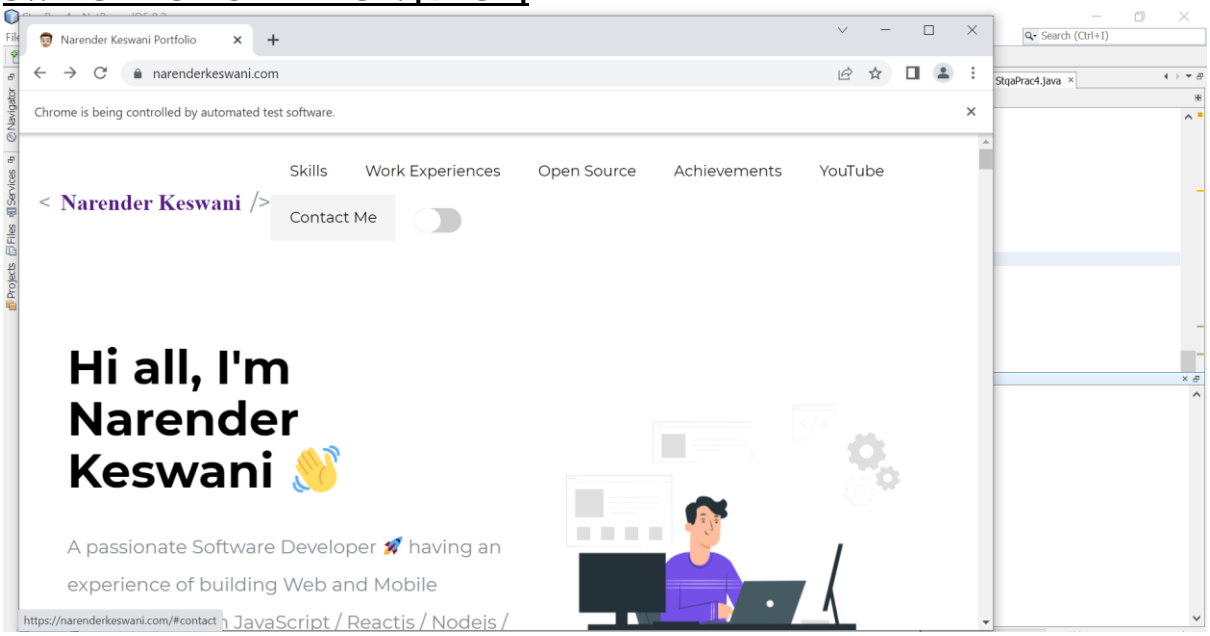
LOADING WEBSITE:



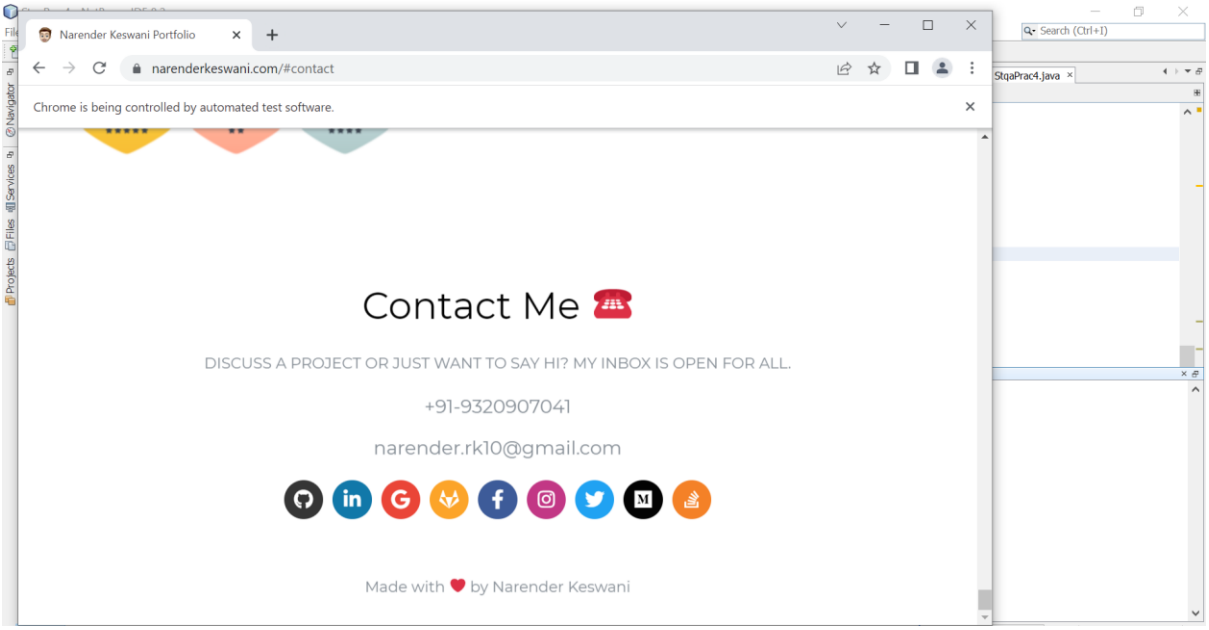
SWITCHING TO CONTACT US:



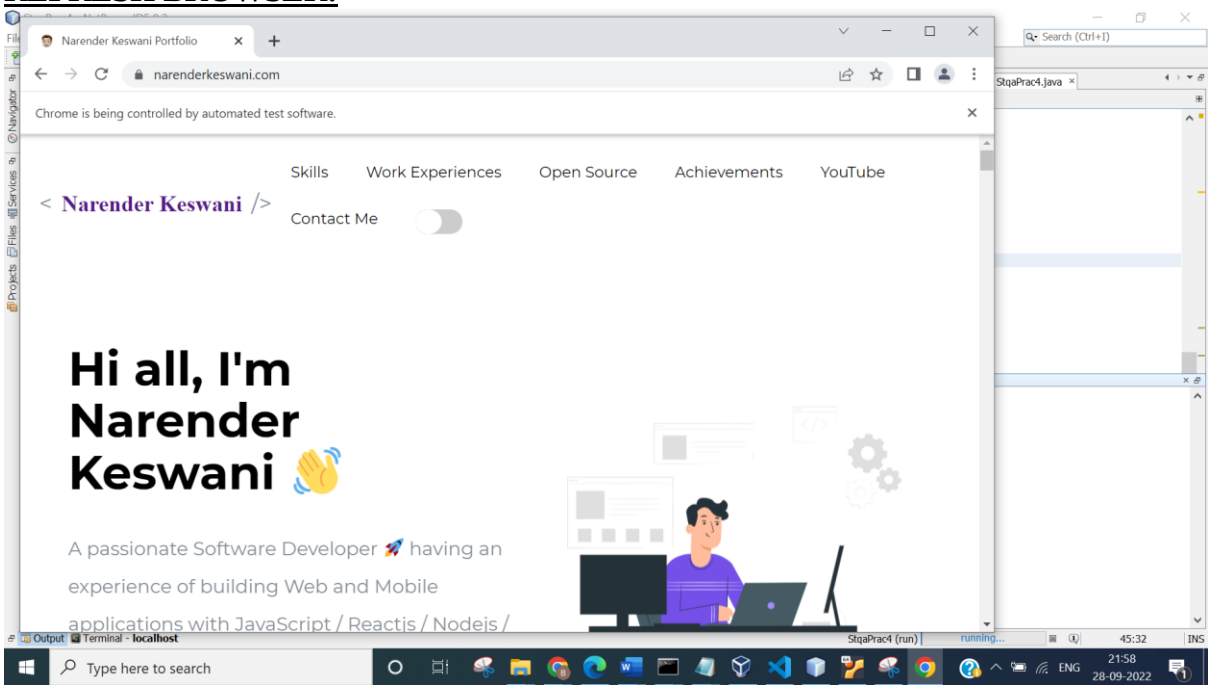
SWITCH TO HOME PAGE: [BACK]



SWITCH TO CONTACT US: [FORWARD]



REFRESH BROWSER:



BROWSER IS CLOSED

```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 105.0.5195.52 (412c95e518836d8a7d97250d62b29c2ae6a26a85-refs/branch-heads/5195@{#853}) on port 64881
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
Sep 28, 2022 9:50:11 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Sep 28, 2022 9:50:12 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 105, so returning the closest version found: 104
Sep 28, 2022 9:50:12 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 105 of 104
BUILD SUCCESSFUL (total time: 24 seconds)
```

CONCLUSION:

Browser and navigation commands are implemented.

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 turn can be used with various locator strategies such as find element by ID Selenium, Name, Class Name, XPATH etc.

Below is the syntax of FindElement command in Selenium web driver:

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

Locator Strategy can be any of the following values.

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

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

FindElements:

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

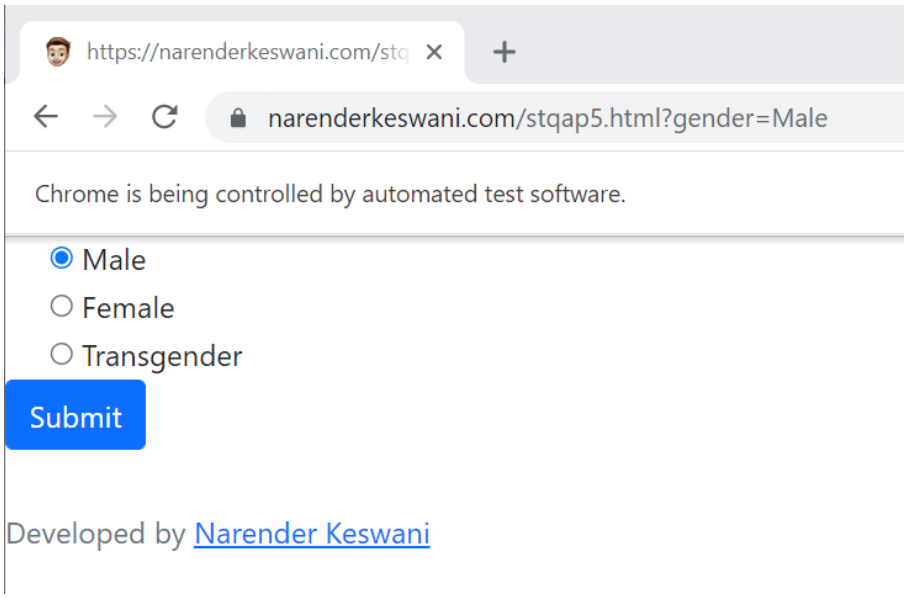
Below is the syntax of find elements command.

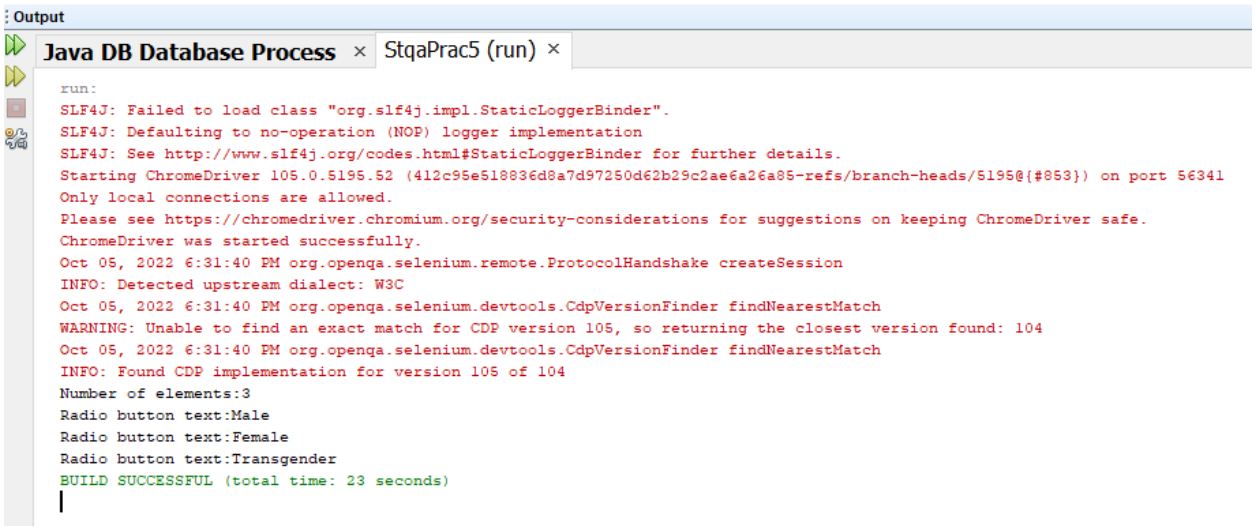
```
List<WebElement> elementName =
driver.findElements(By.LocatorStrategy("LocatorValue"));
```

SOURCE CODE:

```
package stqaprac5;

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

/**
 * @author NARENDER KESWANI
 */
public class StqaPrac5 {
    static WebDriver wd;
    public static void main(String[] args) throws InterruptedException {
        // TODO code application logic here
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
        wd = new ChromeDriver();
        wd.get("https://narenderkeswani.com/stqap5.html");
        Thread.sleep(2000);
        wd.findElement(By.id("flexRadioDefault1")).click();
        Thread.sleep(2000);
        wd.findElement(By.id("buttoncheck")).click();
        Thread.sleep(2000);

        List<WebElement> elements = wd.findElements(By.name("gender"));
        System.out.println("Number of elements:" + elements.size());
        Thread.sleep(2000);
        for (int i = 0; i < elements.size(); i++) {
            System.out.println("Radio button text:" + elements.get(i).getAttribute("value"));
            Thread.sleep(2000);
        }
    }
}
```

OUTPUT:





CONCLUSION:

Find element command has been successfully implemented.

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

THEORY:

What are Locators?

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

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

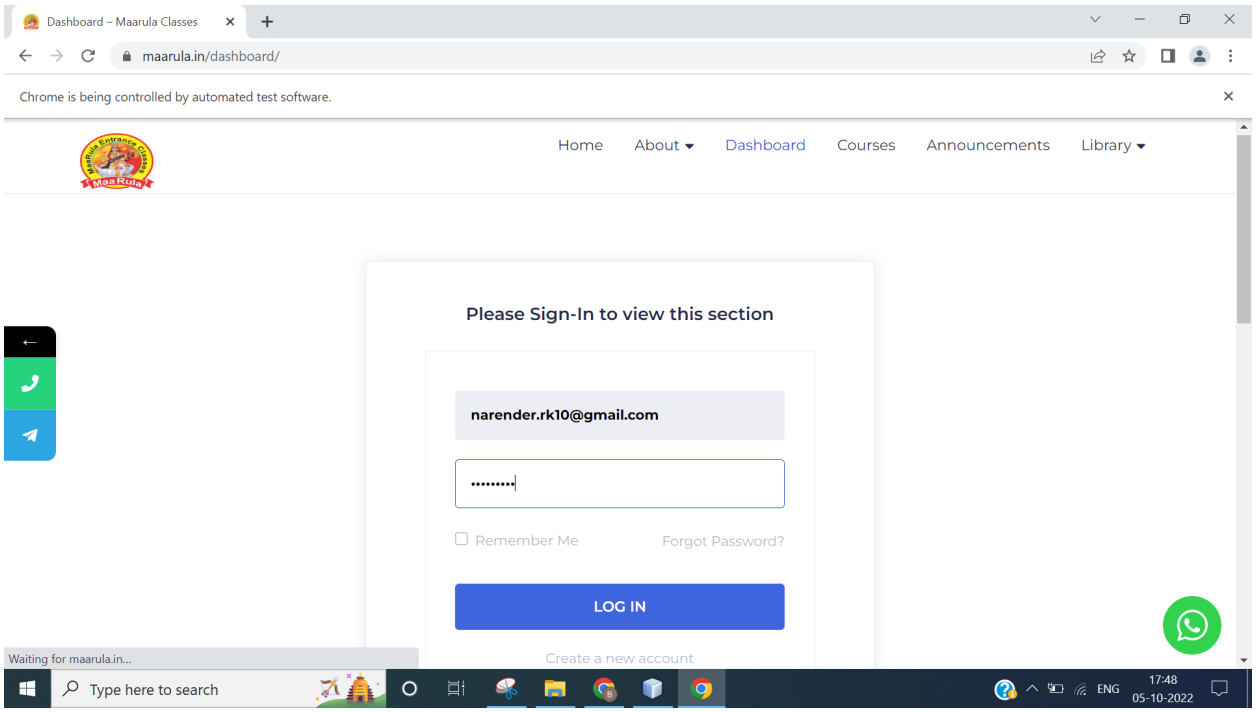
SOURCE CODE:

```
package stqaprac6;

import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
/**
 * @author NARENDER KESWANI
 */
public class StqaPrac6 {
   static WebDriver wd;
   public static void main(String[] args) throws InterruptedException {
      // TODO code application logic here
System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
                    wd = new ChromeDriver();
                    wd.get("https://maarula.in/dashboard/");
                    wd.manage().window().maximize();
                    Thread.sleep(5000);
                    wd.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);
                    wd.findElement(By.name("log")).sendKeys("narender.rk10@gmail.com");
```

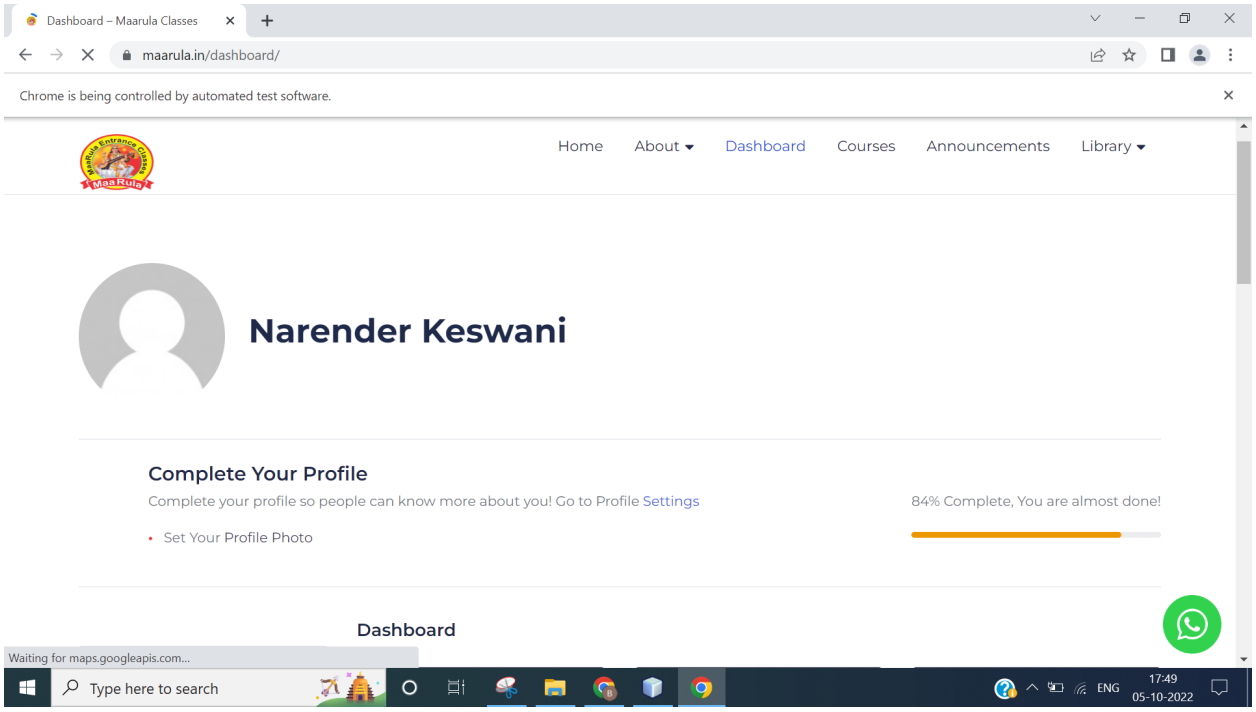
```
                Thread.sleep(2000);
                wd.findElement(By.id("user_pass")).sendKeys("ENTER_YOUR_PASSWORD");
                Thread.sleep(2000);
                wd.findElement(By.id("wp-submit")).click();
                Thread.sleep(2000);
wd.findElement(By.xpath("/html/body/div[1]/div/div/div[1]/div[2]/div/div[2]/div[1]/p/a")).click();
                Thread.sleep(2000);

wd.findElement(By.cssSelector("input[name='phone_number']")).sendKeys("9320907041");
                Thread.sleep(2000);
wd.findElement(By.cssSelector("input[name='phone_number']")).sendKeys(Keys.ENTER);
                Thread.sleep(2000);
wd.navigate().to("https://maarula.in/dashboard/logout");
                Thread.sleep(2000);
    }
}
```

OUTPUT:

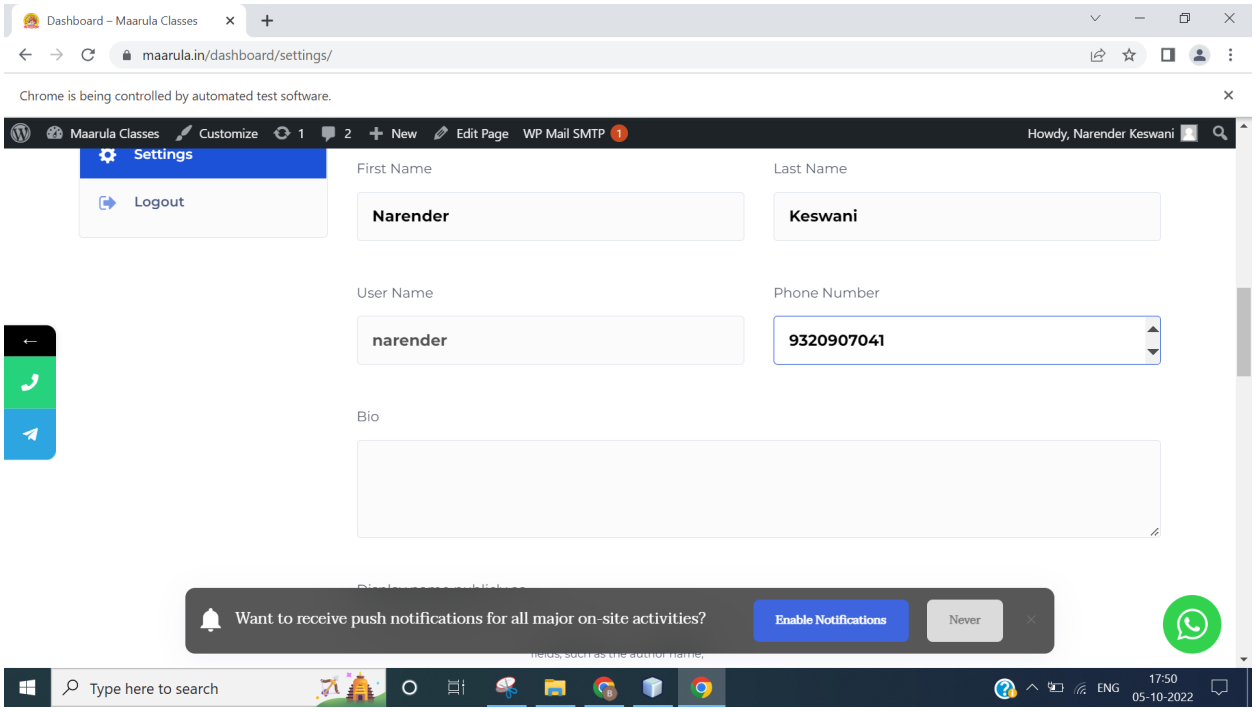
LOGIN WITH EMAIL & PASSWORD:



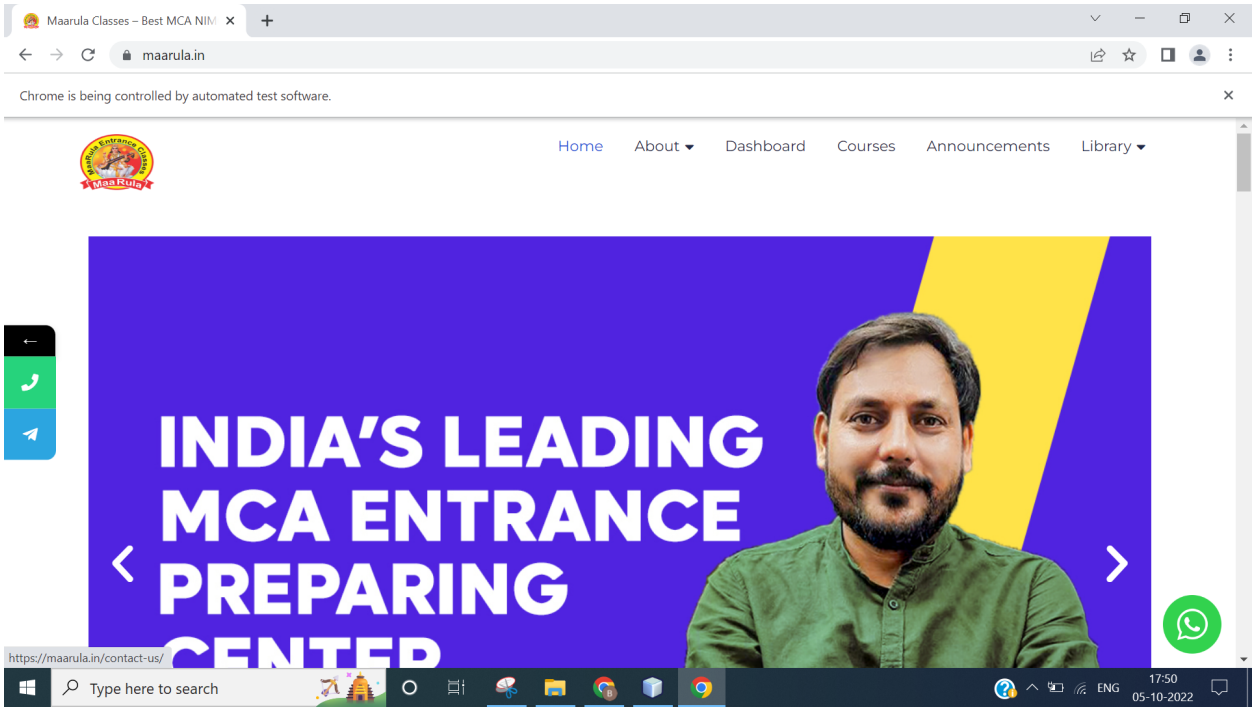
OPENING DASHBOARD:



UPDATING PROFILE: [AUTO FILLING OF MOBILE NUMBER] & CLICKING ON UPDATE PROFILE



LOGOUT & REDIRECTING ON THE HOMEPAGE:



CONCLUSION:

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

Aim: Demonstrate synchronization in selenium

Theory:

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

Thread.Sleep:

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

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

Explicit Waits:

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

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

Implicit wait:

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

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

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

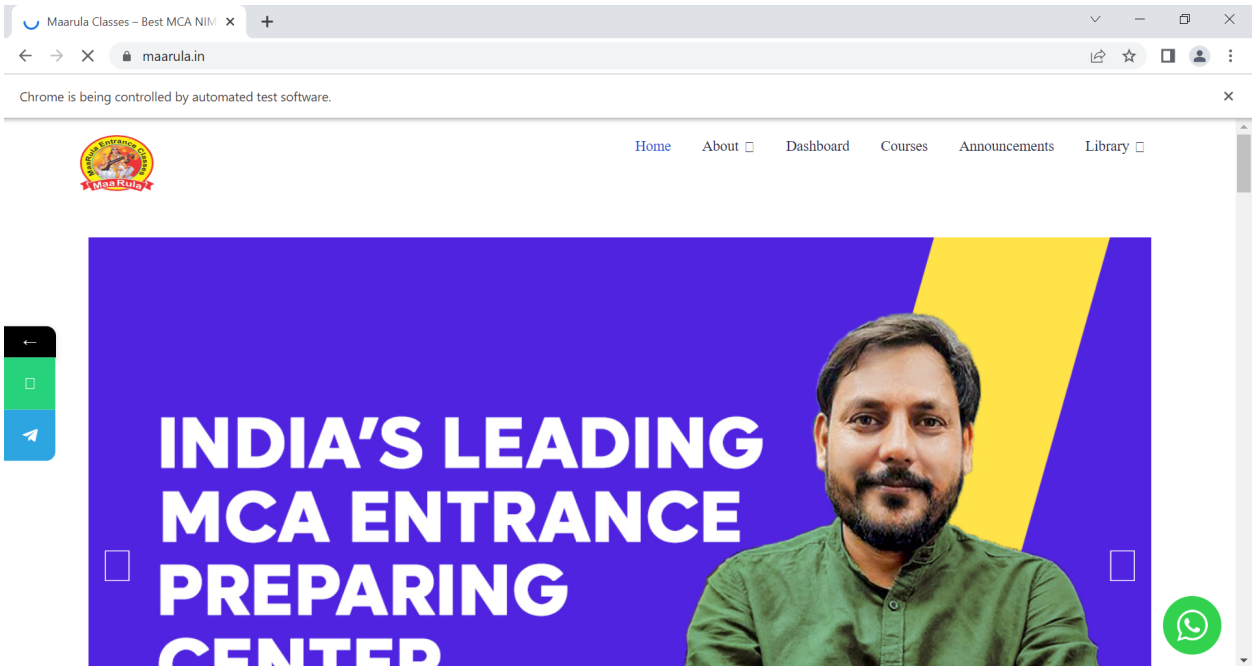
```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("Enter an URL");
```

IMPLICIT WAIT:

SOURCE CODE:

```
package stqaprac7;

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

/**
 * @author NARENDER KESWANI
 */
public class StqaP7A {

    static WebDriver driver;
    public static void main(String[] args) throws InterruptedException {
        // TODO code application logic here
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");

        driver = new ChromeDriver();
        driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
        String eTitle = "Maarula Classes – Best MCA NIMCET Classes";
        String aTitle = "";

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

OUTPUT:




```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 105.0.5195.52 (412c95e518836d8a7d97250d62b29c2ae6a26a85-refs/branch-heads/5195@{#853}) on port 58242
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1665607280.842][WARNING]: This version of ChromeDriver has not been tested with Chrome version 106.
Oct 13, 2022 2:11:21 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 13, 2022 2:11:21 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 13, 2022 2:11:21 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Test Passed
BUILD SUCCESSFUL (total time: 9 seconds)
```

EXPLICIT WAIT:

SOURCE CODE:

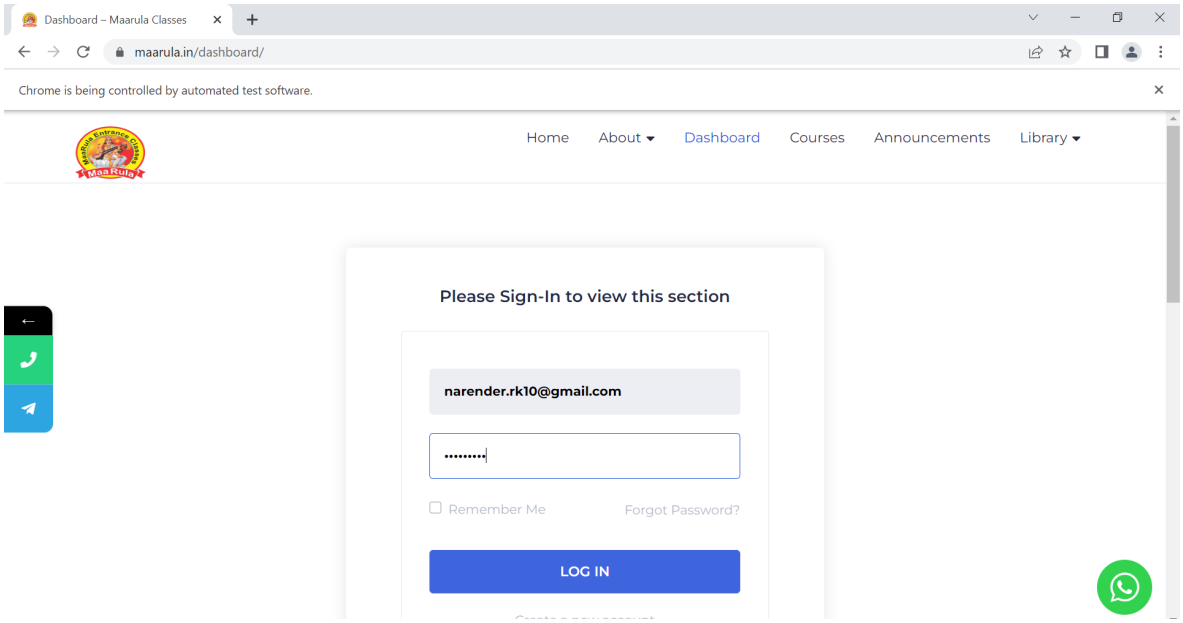
```
package stqaprac7;

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

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

/**
 * @author NARENDER KESWANI
 */
public class StqaP7B {

    static WebDriver driver;

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");

        driver = new ChromeDriver();
        // explicit wait - to wait for the compose button to be click-able
        Duration durationInMinutes = Duration.ofMinutes(30);
        WebDriverWait wait = new WebDriverWait(driver, durationInMinutes);

        // launch Chrome and redirect it to the Base URL
        String eTitle = "Dashboard – Maarula Classes";
        String aTitle = "";
        // launch Chrome and redirect it to the Base URL
        driver.get("https://maarula.in/dashboard/");
        //Maximizes the browser window
        driver.manage().window().maximize();
        //get the actual value of the title
        aTitle = driver.getTitle();
        //compare the actual title with the expected title

        if (aTitle.contentEquals(eTitle)) {
            System.out.println("Test Passed");
        } else {
            System.out.println("Test Failed");
        }

        driver.findElement(By.name("log")).sendKeys("narender.rk10@gmail.com");
        Thread.sleep(2000);

        driver.findElement(By.id("user_pass")).sendKeys(“your_password");
        Thread.sleep(2000);

        driver.findElement(By.id("wp-submit")).click();
        Thread.sleep(2000);
```

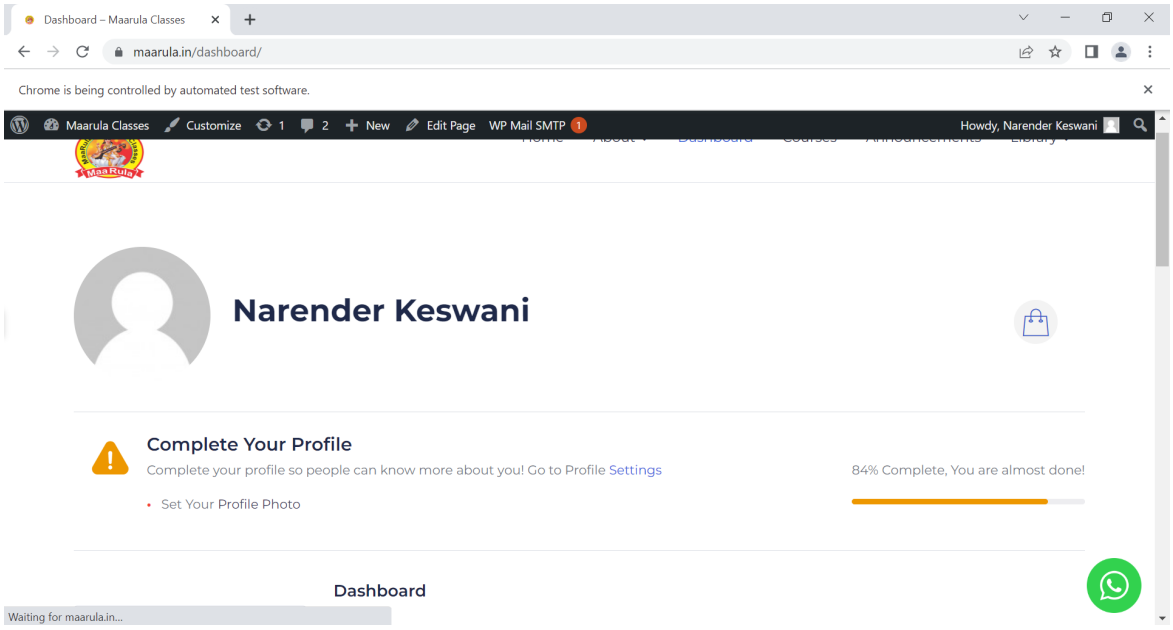
```
        WebElement webElement;
        webElement =
wait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Logout")));
        webElement.click();
    }
}
```

OUTPUT:

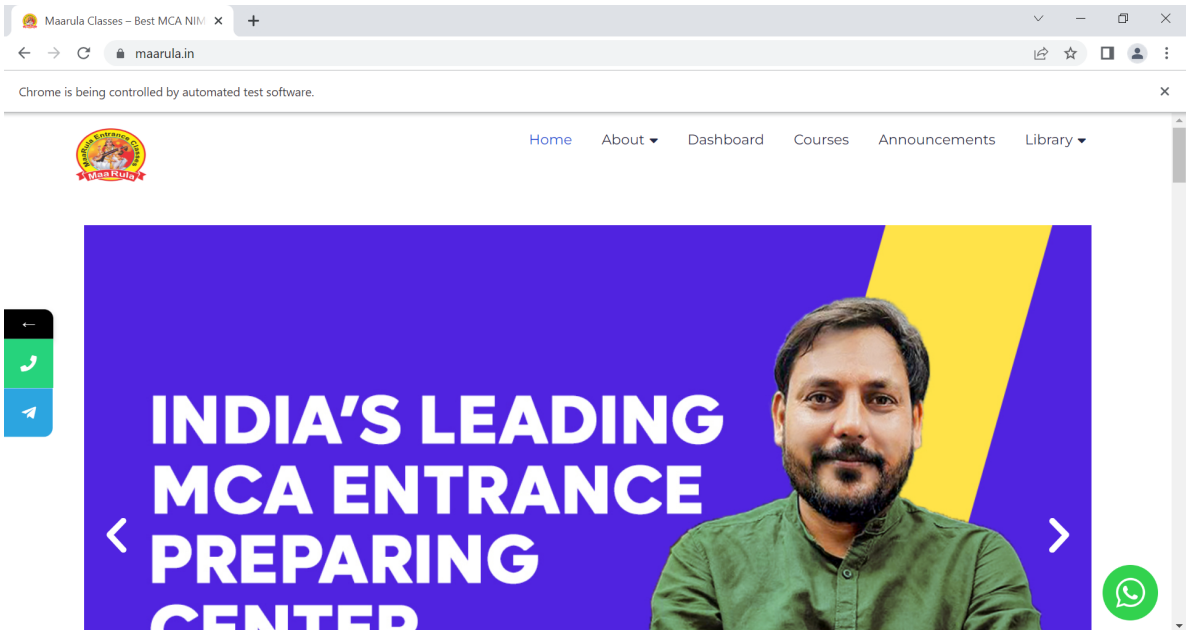
LOGIN:



DASHBOARD:



LOGOUT:



CONCLUSION:

Implicit wait and explicit wait have been demonstrated.

Aim: Demonstrate different types of alerts

THEORY:

What is Alert in Selenium?

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

How to Handle Alert in Selenium WebDriver:

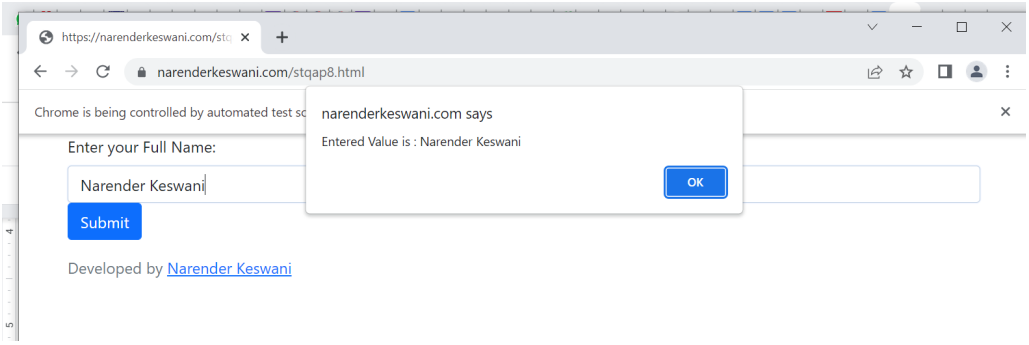
1) **To click on the 'Cancel' button of the alert.**
   void dismiss()
   `driver.switchTo().alert().dismiss();`
2) **To click on the 'OK' button of the alert.**
   void accept()
   `driver.switchTo().alert().accept();`
3) **To capture the alert message.**
   String getText**()**
   `driver.switchTo().alert().getText();`
4) **To send some data to the alert box.**
   void sendKeys(String stringToSend)
   `driver.switchTo().alert().sendKeys("Text");`

SOURCE CODE:

```
package stqaprac8;

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

/**
 * @author NARENDER KESWANI
 */
public class Stqaprac8 {
   static WebDriver wd;
   public static void main(String[] args) throws InterruptedException {
      // TODO code application logic here

      System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
      wd = new ChromeDriver();
```

```
        wd.get("https://narenderkeswani.com/stqap8.html");
        wd.findElement(By.name("fullname")).sendKeys("Narender Keswani");
        wd.findElement(By.name("submit")).click();

        // Switching to Alert
        Alert alert = wd.switchTo().alert();

        // Capturing alert message.
        String alertMessage = wd.switchTo().alert().getText();

        // Displaying alert message
        System.out.println(alertMessage);
        Thread.sleep(2000);
    }
}
```

OUTPUT:




```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 105.0.5195.52 (412c95e518836d8a7d97250d62b29c2ae6a26a85-refs/branch-heads/5195@{#853}) on port 57789
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1665610991.229][WARNING]: This version of ChromeDriver has not been tested with Chrome version 106.
Oct 13, 2022 3:13:11 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 13, 2022 3:13:11 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 13, 2022 3:13:11 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Entered Value is : Narender Keswani
BUILD SUCCESSFUL (total time: 7 seconds)
```

CONCLUSION:

Concept of Alert has been Demonstrated.

Aim:Demonstrate Handling Drop Down and List Boxes.

Theory:

Select Class in Selenium

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

Select Option from Drop-Down Box

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

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

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

ListBox:

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

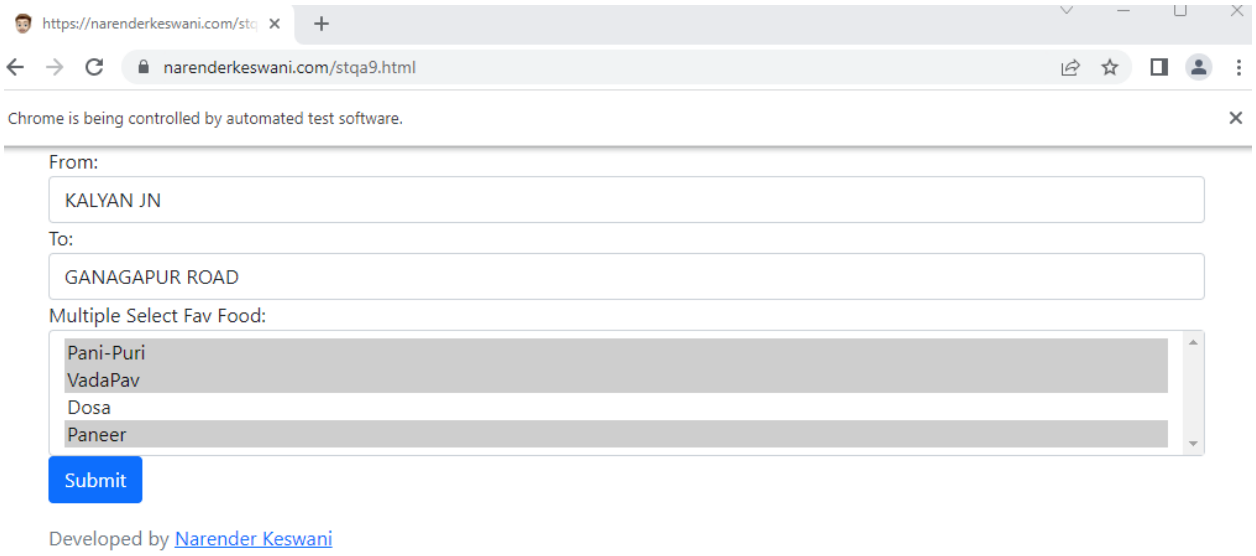
To identify the ListBox on webpage, look for select tag and attribute should be 'multiple' to select multiple items and there will be option tag which contains each item in it.

SOURCE CODE:

```
package stqaprac9;

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

/**
 * @author NARENDER KESWANI
 */
public class StqaPrac9 {
    static WebDriver wd;

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

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

        wd.get("https://narenderkeswani.com/stqa9.html");

        Select s = new Select(wd.findElement(By.name("fromStn")));
        Select t = new Select(wd.findElement(By.name("toStn")));
        s.selectByVisibleText("KALYAN JN");
        t.selectByVisibleText("GANAGAPUR ROAD");

        String sMessage = s.getFirstSelectedOption().getText();
        String tMessage = t.getFirstSelectedOption().getText();

        System.out.println(sMessage);
        System.out.println(tMessage);

        Thread.sleep(2000);

        Select list = new Select(wd.findElement(By.name("favFood")));

        if (list.isMultiple()) {
            list.selectByIndex(0);
            list.selectByValue("vadavpav");
            list.selectByVisibleText("Paneer");
            System.out.println("Selected:");

            List<WebElement> selected1 = list.getAllSelectedOptions();
            for (WebElement el : selected1) {
                System.out.println("Selected: " + el.getAttribute("value"));
            }

            Thread.sleep(5000);
            list.deselectByIndex(3);
            list.deselectAll();
            System.out.println("Deselected");
        }
    }
}
```

OUTPUT:




```
INFO: Found CDP implementation for version 106 of 104
KALYAN JN
GANAGAPUR ROAD
Selected:
Selected: panipuri
Selected: vadavpav
Selected: paneer
Deselected
BUILD SUCCESSFUL (total time: 10 seconds)
```

CONCLUSION:

Concept of DropDown and ListBox has been demonstrated.

AIM: Demonstrate Command Button, Radio buttons & text boxes

THEORY:

Selenium WebDriver - Navigation Commands

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

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

1. **Navigate To Command**

   Method: to(String arg0) : **void**

   In WebDriver, this method loads a new web page in the existing browser window. It accepts *String* as parameter and returns *void*.
   The respective command to load/navigate a new web page can be written as:

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

2. **Forward Command**

   Method: to(String arg0) : **void**

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

   ```
   driver.navigate().forward();
   ```

3. **Back Command:**

   Method: back() : void

   In WebDriver, this method enables the web browser to click on the back button in the existing browser window. It neither accepts anything nor returns anything.
   The respective command that takes you back by one page on the browser's history can be written as: driver.navigate().back();

4. **Refresh Command:**

   Method: refresh() : void

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

Radio Button

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

Input Box

Input boxes refer to either of these two types:

1) **Text Fields**– Selenium input text boxes that accept typed values and show them as they are.
2) **Password Fields**– text boxes that accept typed values but mask them as a series of special characters (commonly dots and asterisks) to avoid sensitive values to be displayed

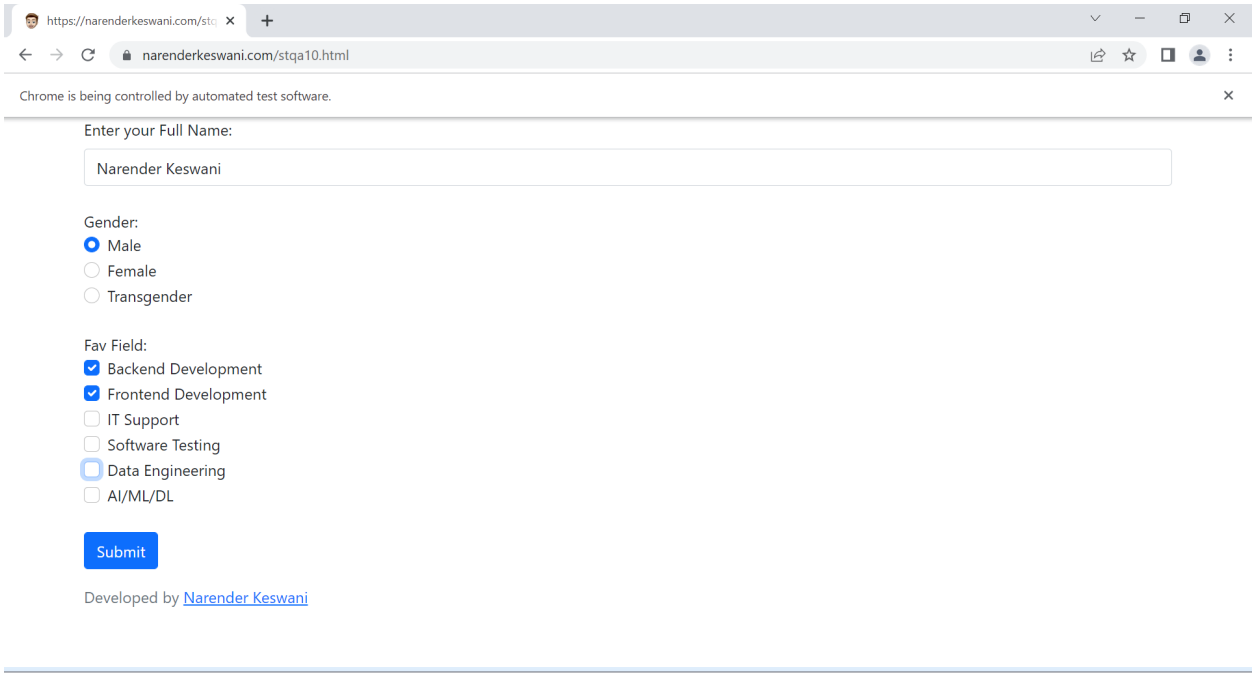
SOURCE CODE:

```
package stqaprac10;

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

/**
 * @author NARENDER KESWANI
 */
public class StqaPrac10 {
   static WebDriver wd;

   public static void main(String[] args) {
      // TODO code application logic here

      System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
      wd = new ChromeDriver();

      wd.get("https://narenderkeswani.com/stqa10.html");

      WebElement l = wd.findElement(By.name("fullname"));
      l.sendKeys("Narender Keswani");
      String val = l.getAttribute("value");
      System.out.println("Entered fullname is: " + val);

      WebElement radio1 = wd.findElement(By.id("flexRadioDefault1"));
      WebElement radio2 = wd.findElement(By.id("flexRadioDefault2"));
      WebElement radio3 = wd.findElement(By.id("flexRadioDefault3"));

      //Radio Button3 is selected
      radio3.click();
      System.out.println("Radio Button Option 3 (transgender) Selected");

      //Radio Button1 is de-selected and Radio Button2 is selected
      radio2.click();
```

```
System.out.println("Radio Button Option 2 (Female) Selected");

//Radio Button2 is de-selected and Radio Button1 is selected
radio1.click();
System.out.println("Radio Button Option 1  (Male) Selected");

// Selecting CheckBox
WebElement option1 = wd.findElement(By.id("flexCheckDefault4"));
WebElement option2 = wd.findElement(By.id("flexCheckDefault5"));
WebElement option3 = wd.findElement(By.id("flexCheckDefault6"));
WebElement option4 = wd.findElement(By.id("flexCheckDefault7"));
WebElement option5 = wd.findElement(By.id("flexCheckDefault8"));

// This will Toggle the Check box
option1.click();

// Check whether the Check box is toggled on
if (option1.isSelected()) {
   System.out.println("Checkbox 1 (Backend) is Toggled On");
} else {
   System.out.println("Checkbox 1 (Backend) is Toggled Off");
}

option2.click();

if (option2.isSelected()) {
   System.out.println("Checkbox 2 (Frontend) is Toggled On");
} else {
   System.out.println("Checkbox 2 (Frontend) is Toggled Off");
}

option5.click();

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

option5.click();

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

```
            System.out.println("Checkbox 5 (Data Engineering) is Toggled Off");
        }
    }
}
```

OUTPUT:




```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 105.0.5195.52 (412c95e518836d8a7d97250d62b29c2ae6a26a85-refs/branch-heads/5195@{#853}) on port 53967
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1666198609.085][WARNING]: This version of ChromeDriver has not been tested with Chrome version 106.
Oct 19, 2022 10:26:49 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 19, 2022 10:26:49 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 19, 2022 10:26:49 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Entered fullname is: Narender Keswani
Radio Button Option 3 (transgender) Selected
Radio Button Option 2 (Female) Selected
Radio Button Option 1  (Male) Selected
Checkbox 1 (Backend) is Toggled On
Checkbox 2 (Frontend) is Toggled On
Checkbox 5 (Data Engineering) is Toggled On
Checkbox 5 (Data Engineering) is Toggled Off
BUILD SUCCESSFUL (total time: 10 seconds)
```

CONCLUSION:

Concept of CheckBox and Radio Button has been demonstrated.

Aim: Demonstrate action classes in Selenium

THEORY:

What is Action Class in Selenium?

Actions class is an ability provided by Selenium for handling keyboard and mouse events. In Selenium WebDriver, handling these events includes operations such as drag and drop, clicking on multiple elements with the control key, among others.
These operations are performed using the advanced user interactions API. It mainly consists of *Actions* that are needed while performing these operations.

Action class is defined and invoked using the following syntax:

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

Methods of Action Class:

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

Mouse Actions in Selenium:

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

Keyboard Actions in Selenium:

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

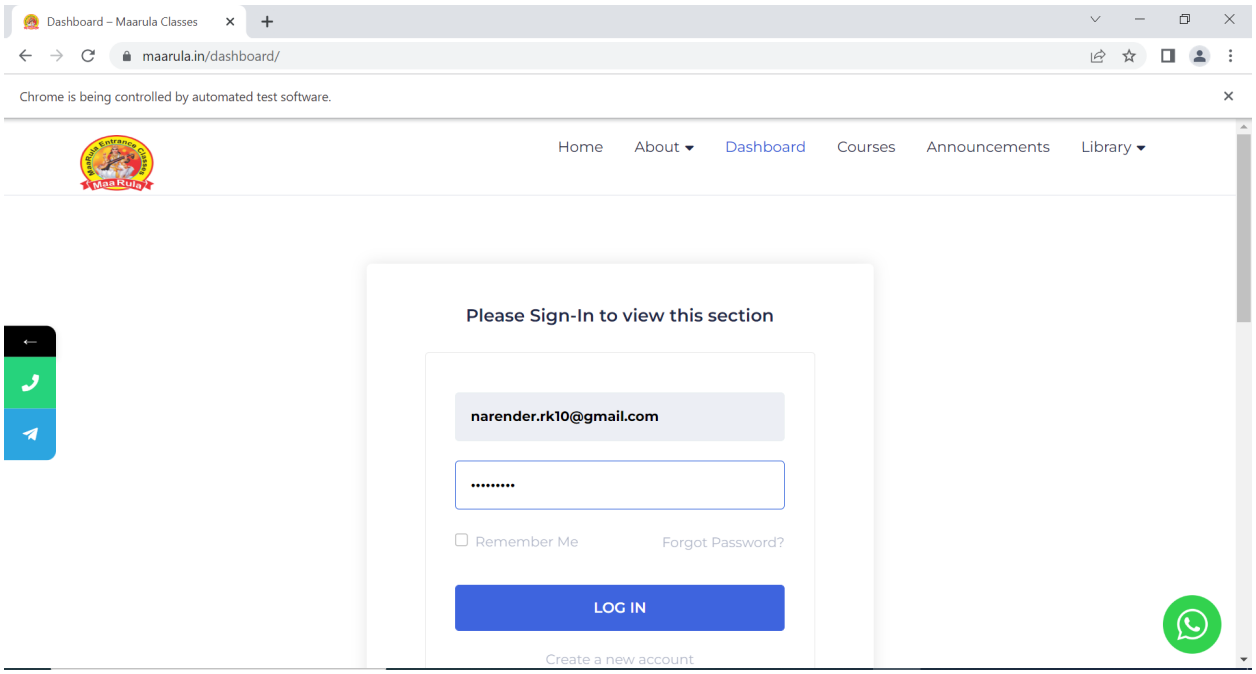
SOURCE CODE:

```
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;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;

/**
 * @author NARENDER KESWANI
 */
public class StqaPrac11 {
```

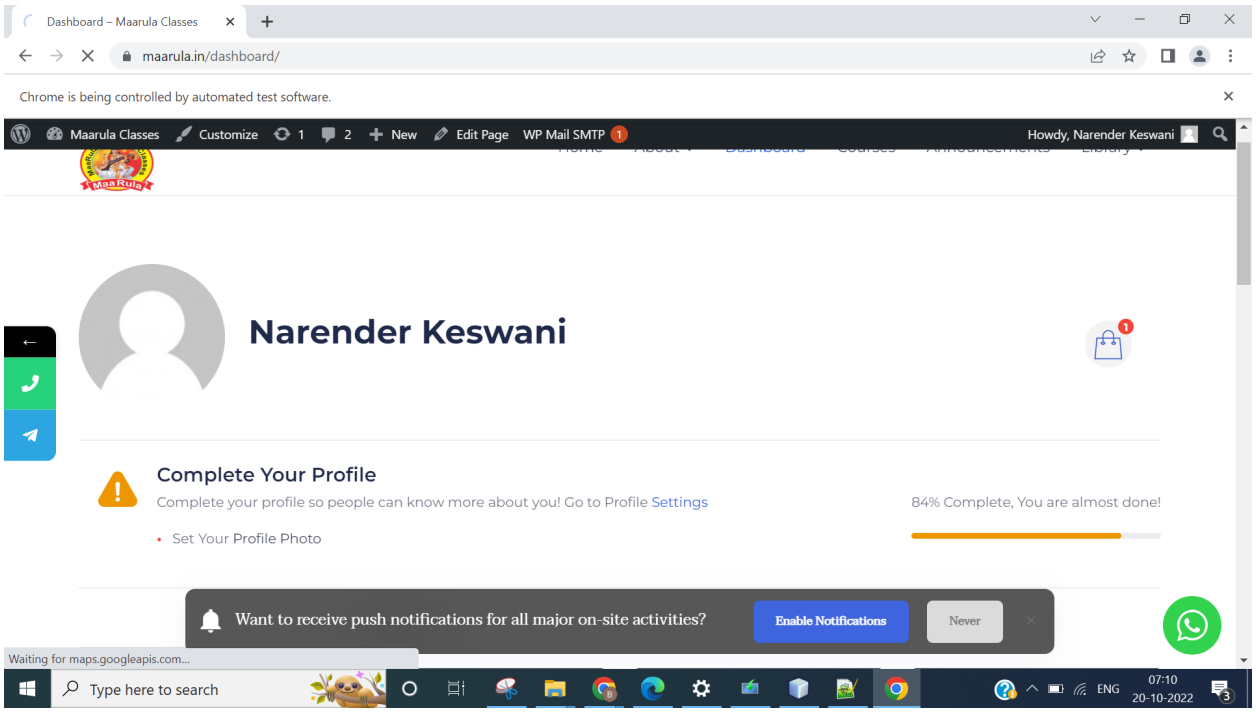
```
   static WebDriver wd;
   public static void main(String[] args) throws InterruptedException {
      // TODO code application logic here

      System.setProperty("webdriver.chrome.driver", "C:\\Users\\NARENDER
KESWANI\\Downloads\\chromedriver_win32\\chromedriver.exe");
      wd = new ChromeDriver();

      wd.get("https://maarula.in/dashboard/");
      wd.manage().window().maximize();
      wd.findElement(By.name("log")).sendKeys("narender.rk10@gmail.com");
      Thread.sleep(2000);

      wd.findElement(By.id("user_pass")).sendKeys("enter_your_password");
      Thread.sleep(2000);

      wd.findElement(By.id("wp-submit")).click();
      Thread.sleep(2000);

      System.out.println("Logged in");

      Actions act = new Actions(wd);
      List<WebElement> menu =
wd.findElements(By.xpath("/html/body/div[1]/header/div[1]/div/div/div/div/div[4]/div/div/ul"));

      System.out.println("Menu List");
      for (int i = 0; i <= menu.size() - 1; i++) {
         System.out.println(menu.get(i).getText() + "\n");
         //print text of all the element on console
         act.moveToElement(menu.get(i)).click();
         //to perform mousehover on all elements of list
         Thread.sleep(2000);
      }
      wd.navigate().to("https://maarula.in/dashboard/logout");
      Thread.sleep(2000);

      System.out.println("Logout");
      wd.close();
   }
}
```

OUTPUT:

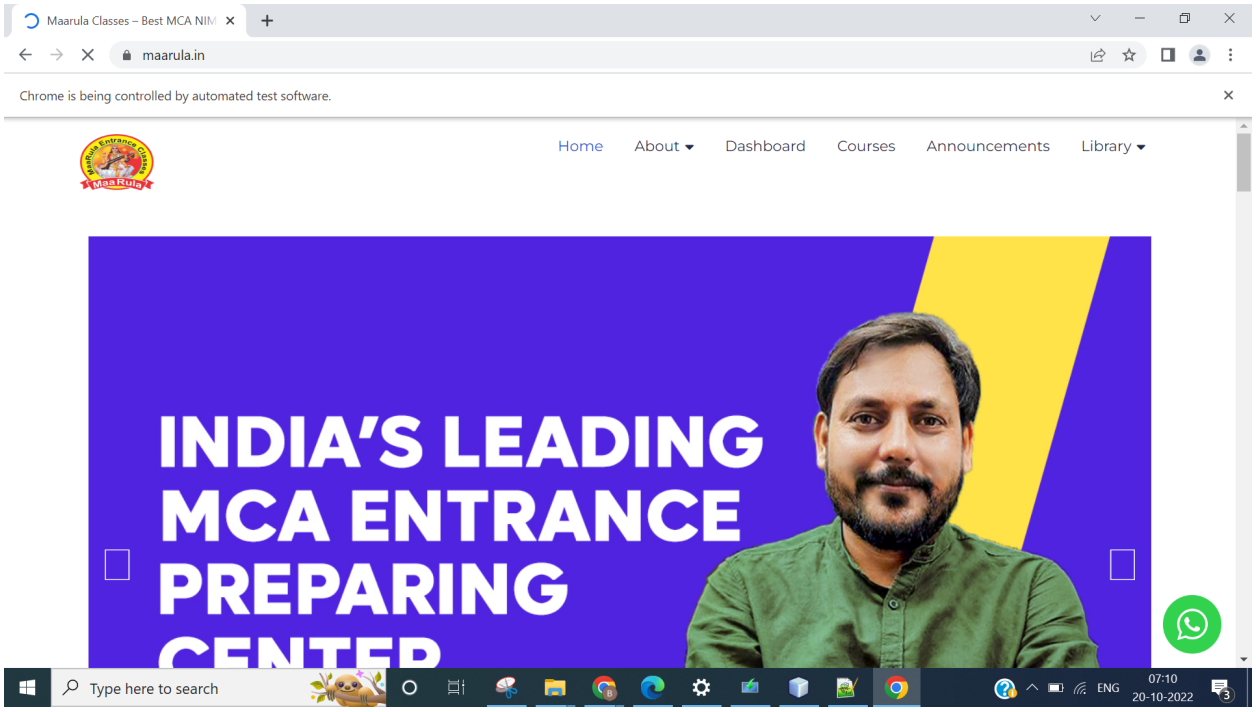
Login:



Get Menu List & Mouse Hover:



Logout:



Printing logs:

```
run:
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Starting ChromeDriver 105.0.5195.52 (412c95e518836d8a7d97250d62b29c2ae6a26a85-refs/branch-heads/5195@{#853}) on port 61197
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
[1666230033.836][WARNING]: This version of ChromeDriver has not been tested with Chrome version 106.
Oct 20, 2022 7:10:34 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected upstream dialect: W3C
Oct 20, 2022 7:10:34 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
WARNING: Unable to find an exact match for CDP version 106, so returning the closest version found: 104
Oct 20, 2022 7:10:34 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found CDP implementation for version 106 of 104
Logged in
Menu List
Home About
Dashboard Courses Announcements Library

Logout
BUILD SUCCESSFUL (total time: 28 seconds)
```

CONCLUSION:

Concept of Action Classes has been demonstrated.

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

THEORY:

What is TestNG?

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

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

For example:

- Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method fails. Then correct the errors present in the fourth method, now you want to run only the fourth method because the first three methods are anyway executed successfully. This is not possible without using TestNG.

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

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

SOURCE 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:

```
<terminated> Stqa12 [TestNG] C:\Users\user\.p2\pool\plugins\org.eclipse.justj.openjd
[RemoteTestNG] detected TestNG version 7.4.0
Before Suite
Before Test
Before Class
Before Method
Test 1
After Method
Before Method
Test 2
After Method
After Class
After Test
PASSED: f1
PASSED: f

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

After Suite

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

CONCLUSION:

Concept of TestNg has been demonstrated.

Aim:Demonstrate data driven Framework.

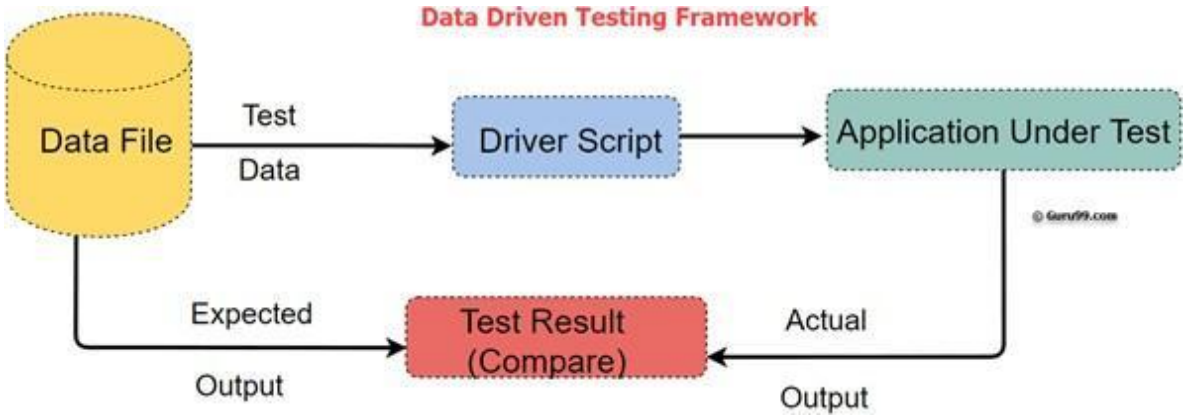
Theory

Data Driven Testing:

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

Data Driven Framework:

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



SOURCE CODE:

Prac13.java:

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.testng.Assert;
import org.testng.annotations.AfterMethod;
import org.testng.annotations.DataProvider;
```

```
import org.testng.annotations.Test;

public class Prac13 {

	WebDriver wd;

	@Test(dataProvider = "testdata")
	public void demoClass(String email, String password) throws InterruptedException {

		System.setProperty("webdriver.chrome.driver",

"C:\\Users\\user\\Downloads\\chromedriver_win321\\chromedriver.exe");
		wd = new ChromeDriver();

		wd.get("https://maarula.in/dashboard/");

		wd.manage().window().maximize();
		Thread.sleep(5000);

		wd.findElement(By.name("log")).sendKeys(email);
		Thread.sleep(2000);

		wd.findElement(By.id("user_pass")).sendKeys(password);
		Thread.sleep(2000);

		wd.findElement(By.id("wp-submit")).click();
		Thread.sleep(2000);

		String str =
wd.findElement(By.xpath("/html/body/div[1]/div/div/div[2]/div[2]/div/div/div/div[3]/p/span[1]")).getT
ext();
		if(str=="Completed Courses")
		{
			System.out.println("Login Successful");
		}
		else
		{
			System.out.println("Login Failed");
		}

	}

	@AfterMethod
	void ProgramTermination() {
```

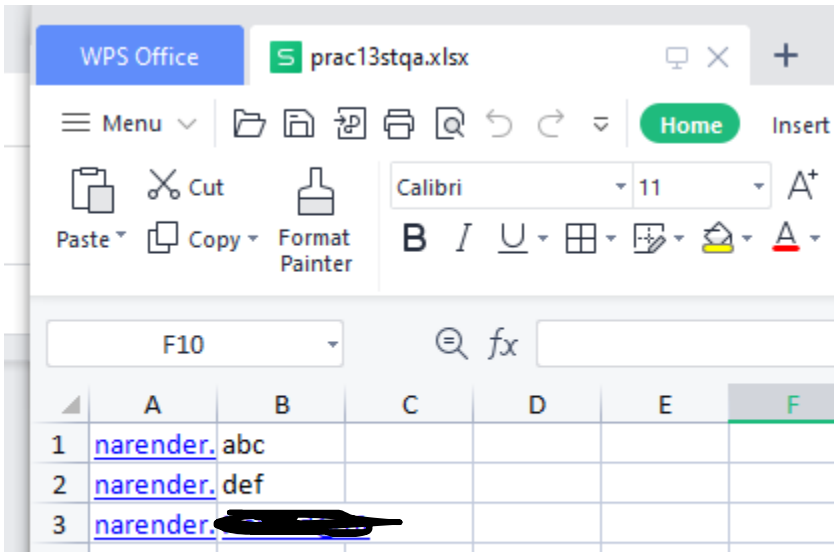
```
		wd.quit();
	}

	@DataProvider(name = "testdata")
	public Object[][] testDataExample() {
		ReadExcelFile configuration = new
ReadExcelFile("C:\\Users\\user\\Documents\\prac13stqa.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:

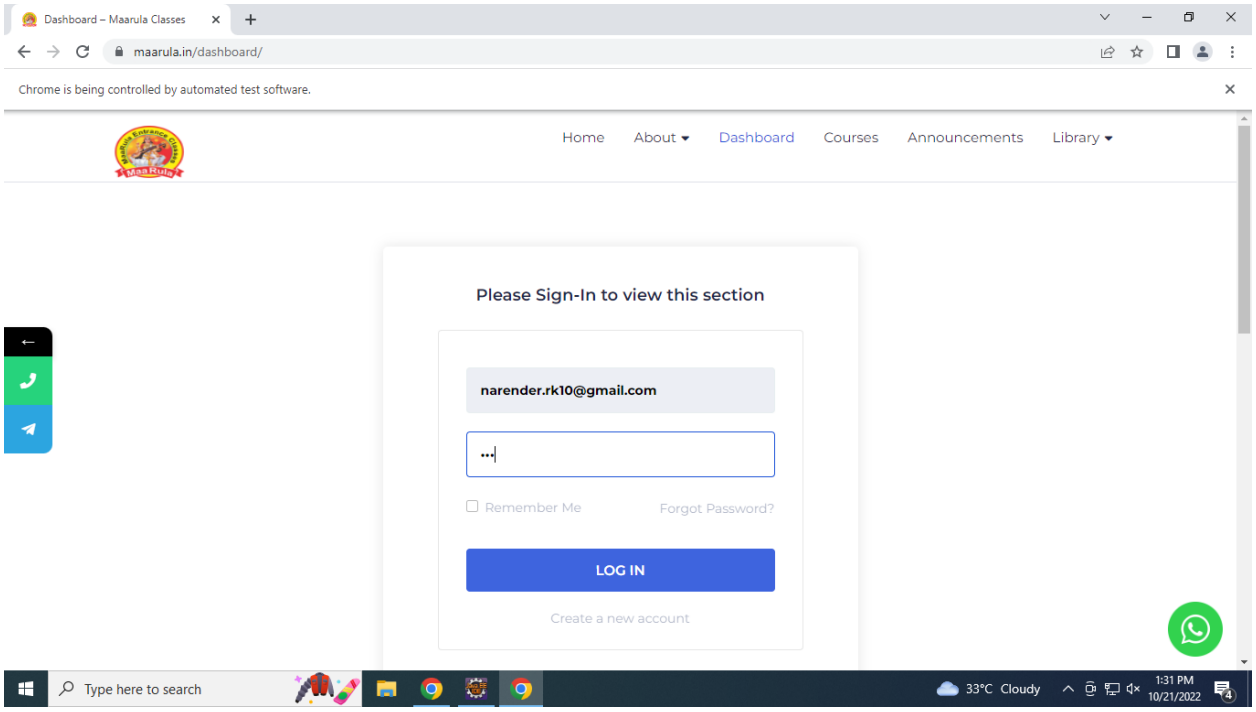
```
import java.io.File;
import java.io.FileInputStream;

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) {
```

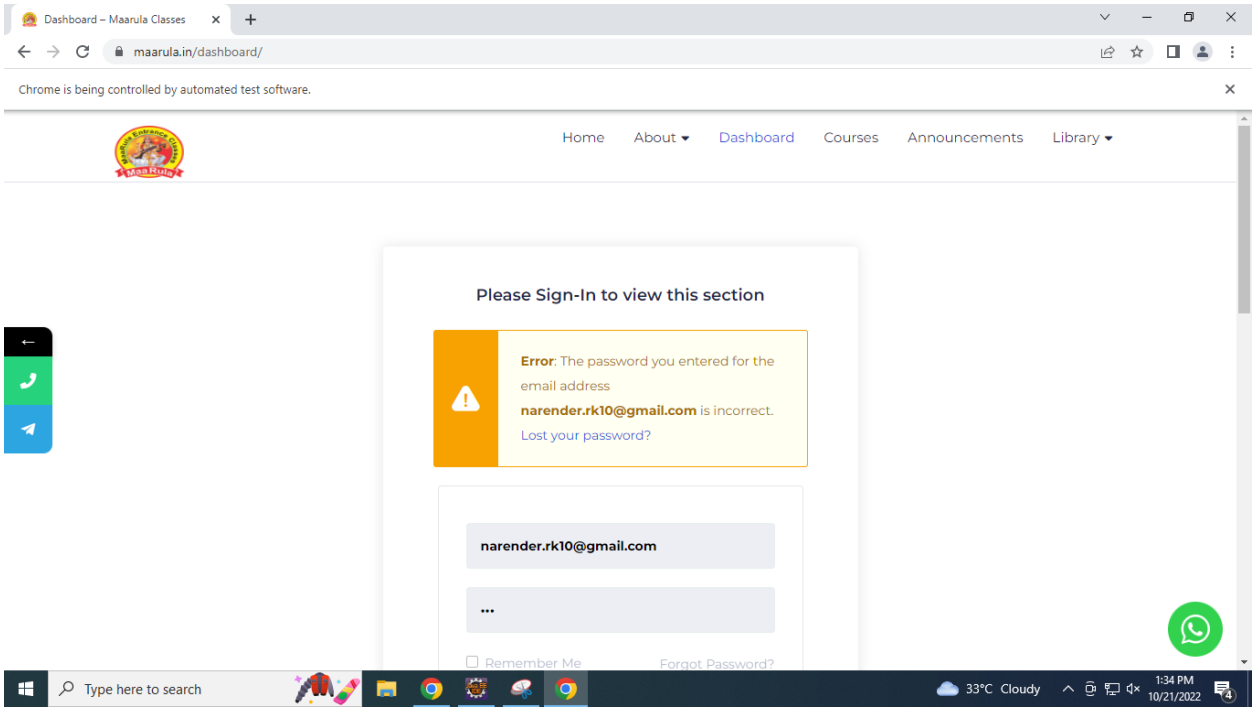
```
            sheet = work_book.getSheetAt(sheetnumber);
            String data = sheet.getRow(row).getCell(column).getStringCellValue();
            return data;
        }

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

OUTPUT:

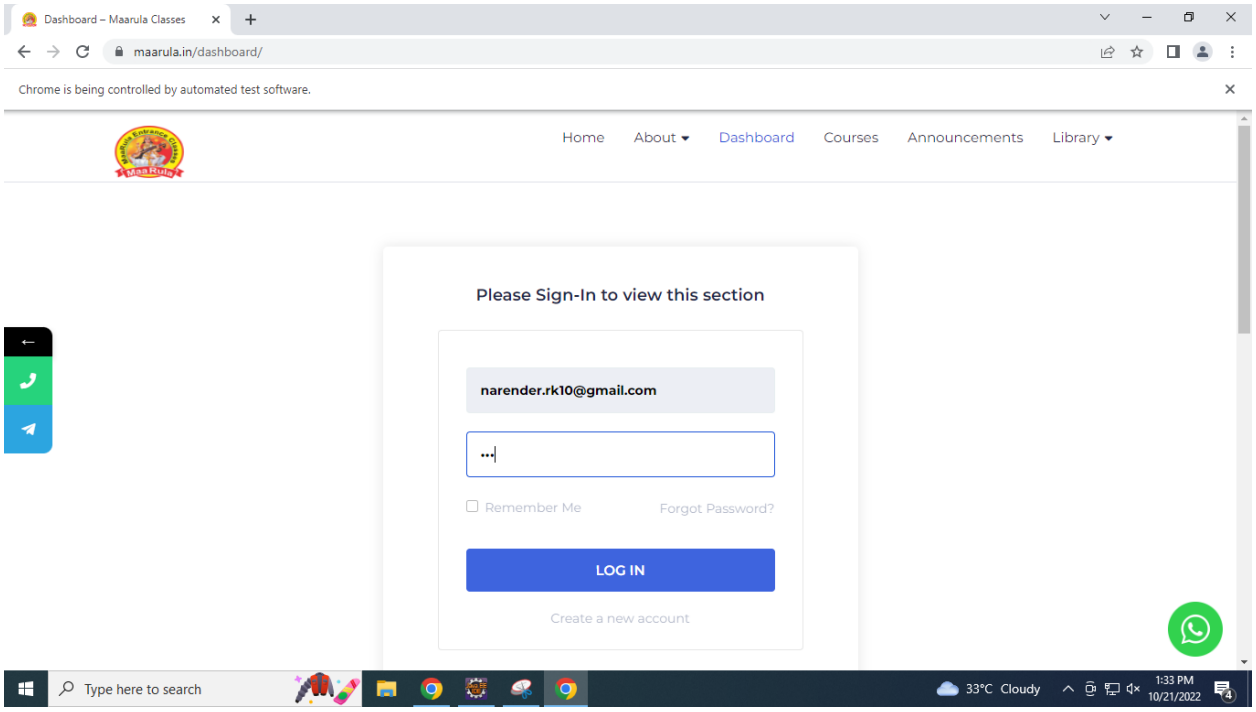


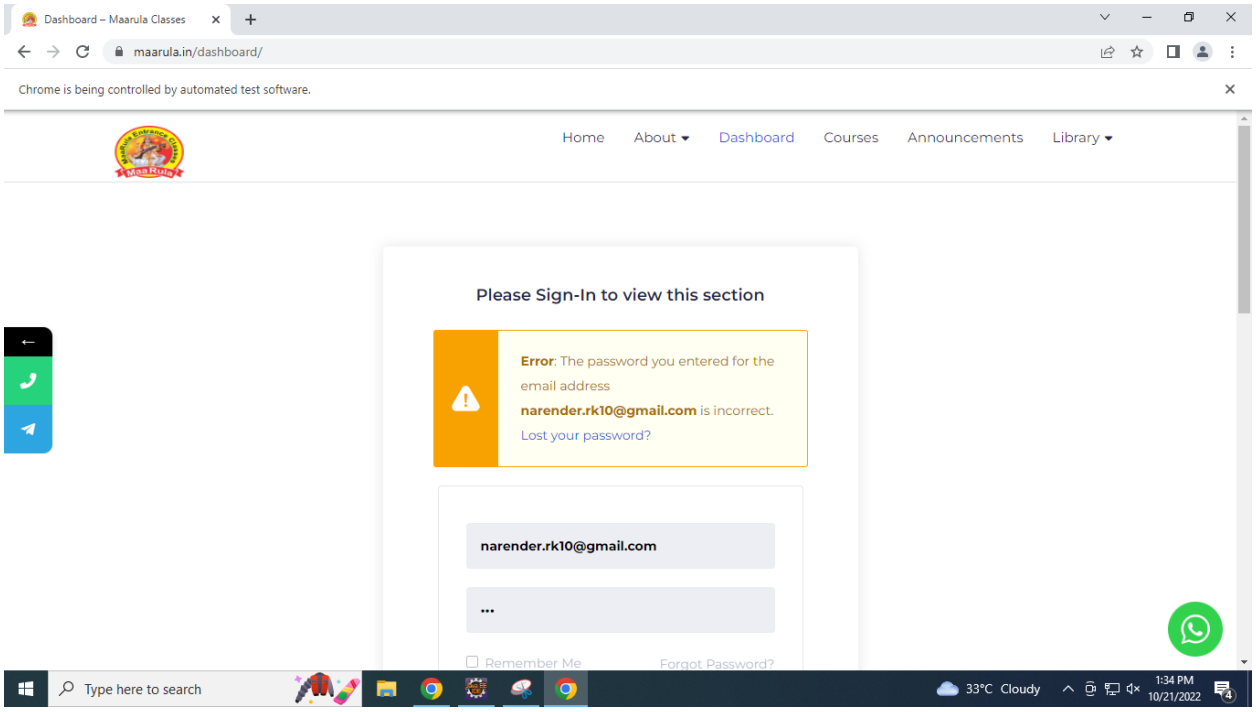
Try1: [failed]


Dashboard – Maarula Classes
maarula.in/dashboard/
Chrome is being controlled by automated test software.
Home
About
Dashboard
Courses
Announcements
Library
Please Sign-In to view this section
narender.rk10@gmail.com
Remember Me
Forgot Password?
LOG IN
Create a new account
Type here to search
33°C Cloudy
1:31 PM
10/21/2022

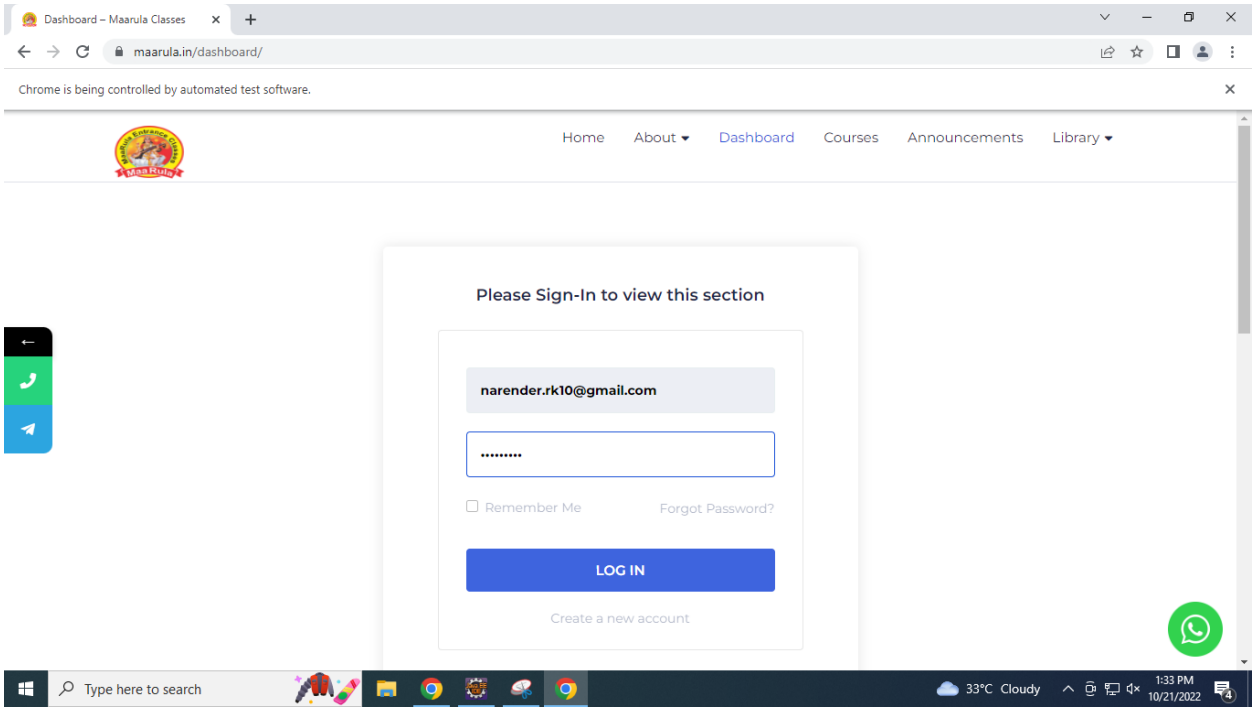

Dashboard – Maarula Classes
maarula.in/dashboard/
Chrome is being controlled by automated test software.
Home
About
Dashboard
Courses
Announcements
Library
Please Sign-In to view this section
Error: The password you entered for the email address narender.rk10@gmail.com is incorrect.
Lost your password?
narender.rk10@gmail.com
Remember Me
Forgot Password?
Type here to search
33°C Cloudy
1:34 PM
10/21/2022

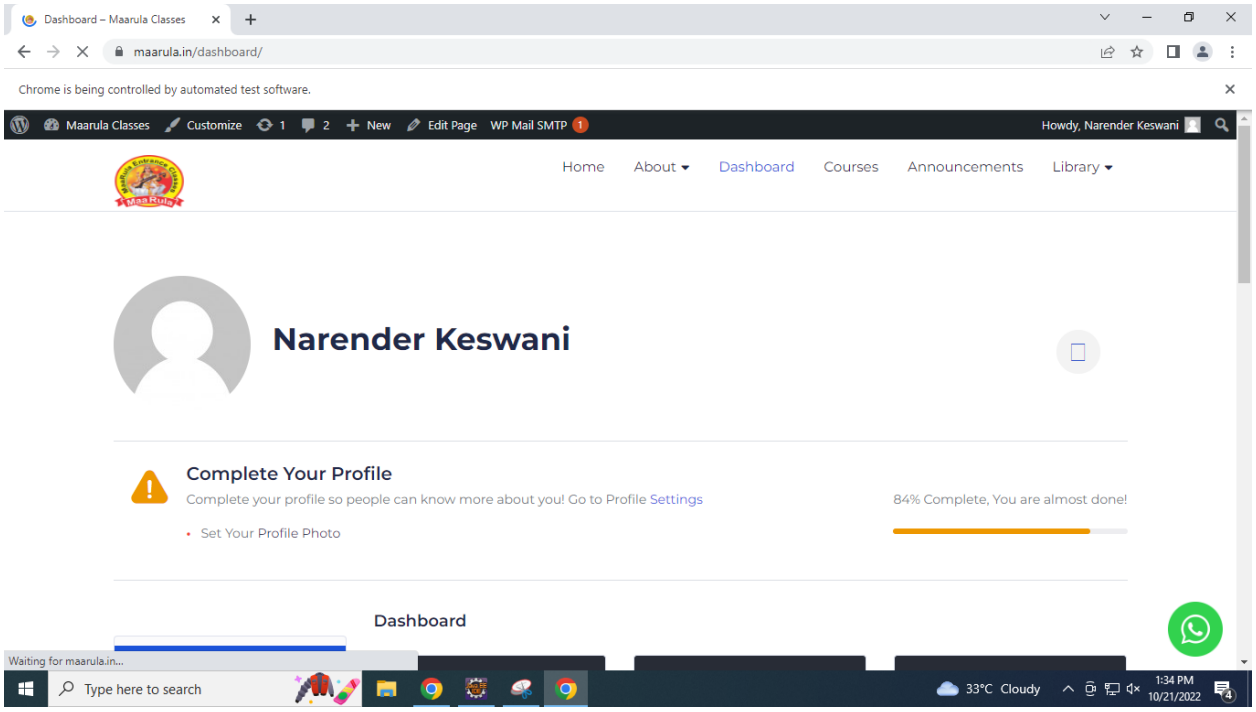
Try 2: [failed]


Dashboard – Maarula Classes
maarula.in/dashboard/
Chrome is being controlled by automated test software.
Home
About
Dashboard
Courses
Announcements
Library
Please Sign-In to view this section
narender.rk10@gmail.com
Remember Me
Forgot Password?
LOG IN
Create a new account
Type here to search
33°C Cloudy
1:33 PM
10/21/2022


Dashboard – Maarula Classes
maarula.in/dashboard/
Chrome is being controlled by automated test software.
Home
About
Dashboard
Courses
Announcements
Library
Please Sign-In to view this section
Error: The password you entered for the email address narender.rk10@gmail.com is incorrect.
Lost your password?
narender.rk10@gmail.com
Remember Me
Forgot Password?
Type here to search
33°C Cloudy
1:34 PM
10/21/2022

Try 3: [success]





Stats: Test ng:

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


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

CONCLUSION:

Concept of a Data Driven Framework has been demonstrated.

A)Software Testing

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Why Software Testing is Important?

Software Testing is Important because if there are any bugs or errors in the software, it can be identified early and can be solved before delivery of the software product. Properly tested software product ensures reliability, security and high performance which further results in time saving, cost effectiveness and customer satisfaction

What are the benefits of Software Testing?

Here are the benefits of using software testing:

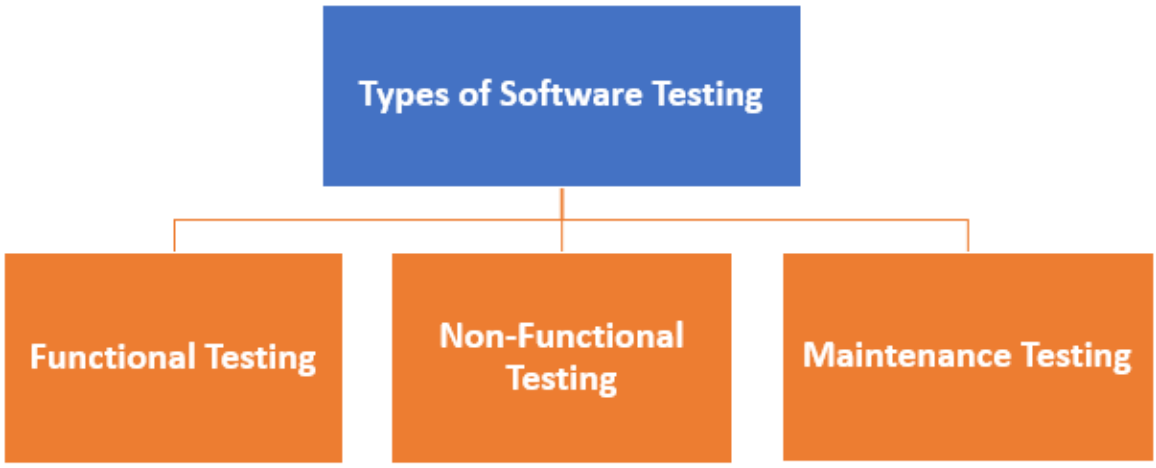
- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Types of Software Testing

Here are the software testing types:

Typically Testing is classified into three categories.

- Functional Testing
- Non-Functional Testing or Performance Testing
- Maintenance (Regression and Maintenance)



Types of Software Testing in Software Engineering

Testing Category	Types of Testing
Functional Testing	• Unit Testing • Integration Testing • Smoke • UAT (User Acceptance Testing) • Localization • Globalization • Interoperability • So on
Non-Functional Testing	• Performance • Endurance • Load • Volume • Scalability • Usability • So on
Maintenance	• Regression • Maintenance

B) What is Functional Testing?

FUNCTIONAL TESTING is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

What do you test in Functional Testing?

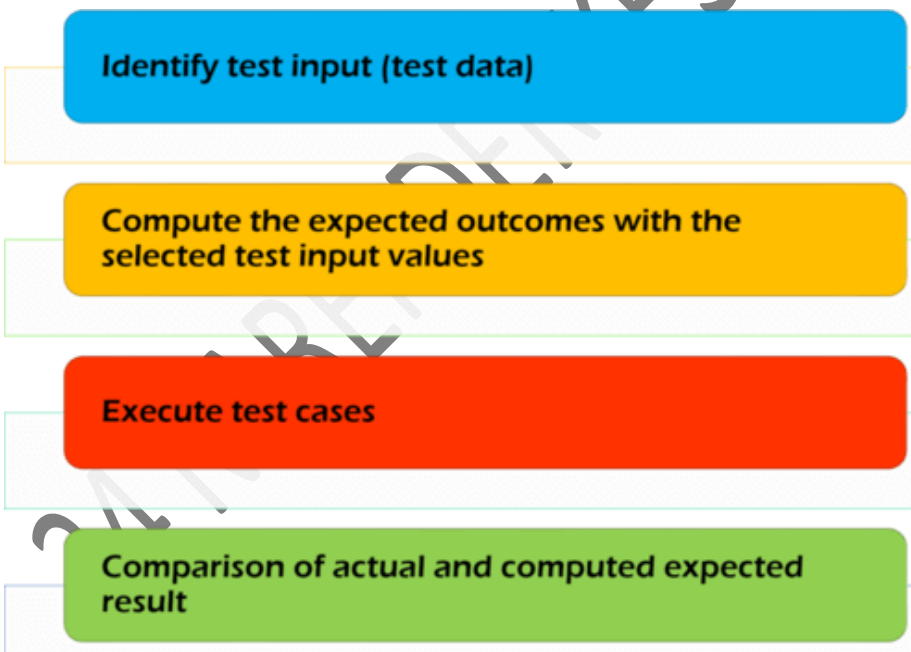
The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on –

- **Mainline functions**: Testing the main functions of an application
- **Basic Usability**: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
- **Accessibility**: Checks the accessibility of the system for the user
- **Error Conditions**: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

How to do Functional Testing

Following is a step by step process on **How to do Functional Testing** :

- Understand the Functional Requirements
- Identify test input or test data based on requirements
- Compute the expected outcomes with selected test input values
- Execute test cases
- Compare actual and computed expected results



C) What is Non-Functional Testing?

Non-Functional Testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to

test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

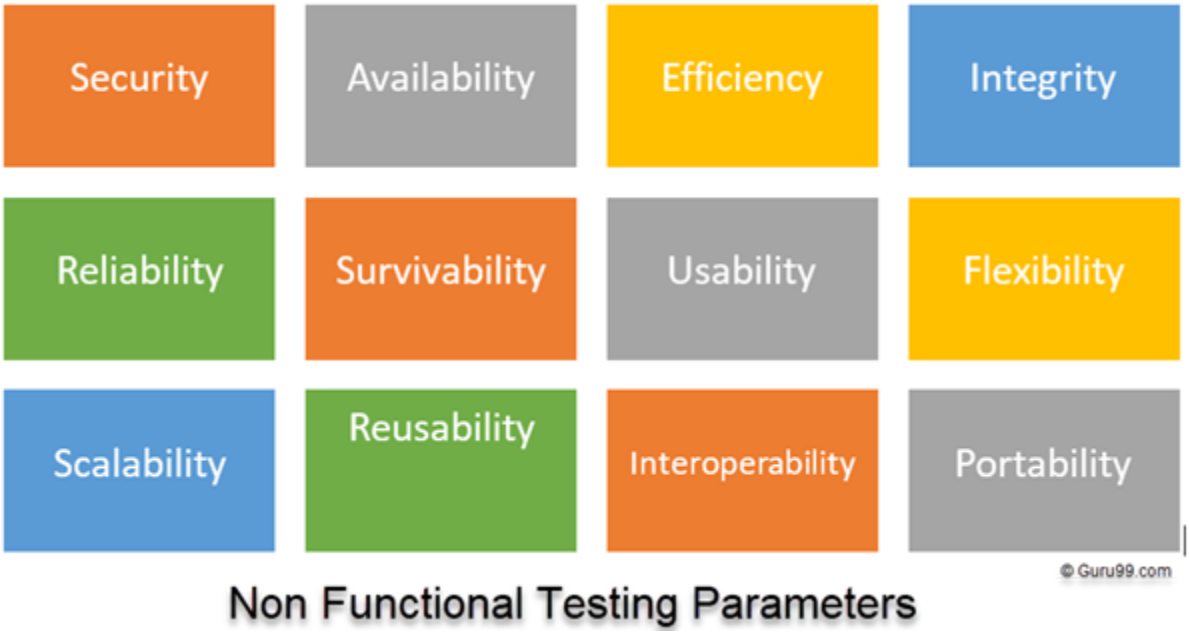
Objectives of Non-functional testing

- Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
- Helps to reduce production risk and cost associated with non-functional aspects of the product.
- Optimize the way product is installed, setup, executes, managed and monitored.
- Collect and produce measurements, and metrics for internal research and development.
- Improve and enhance knowledge of the product behavior and technologies in use.

Characteristics of Non-functional testing

- Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
- Exact numbers are unlikely to be known at the start of the requirement process
- Important to prioritize the requirements
- Ensure that quality attributes are identified correctly in Software Engineering.

Non-functional testing Parameters



Non Functional Testing Parameters

1) Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

2) Reliability:

The extent to which any software system continuously performs the specified functions without failure. This is tested by Reliability Testing

3) Survivability:

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by Recovery Testing

4) Availability:

The parameter determines the degree to which user can depend on the system during its operation. This is checked by Stability Testing.

5) Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by Usability Testing

6) Scalability:

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by Scalability Testing

7) Interoperability:

This non-functional parameter checks a software system interfaces with other software systems. This is checked by Interoperability Testing

8) Efficiency:

The extent to which any software system can handles capacity, quantity and response time.

9) Flexibility:

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

10) Portability:

The flexibility of software to transfer from its current hardware or software environment.

11) Reusability:

It refers to a portion of the software system that can be converted for use in another application.

D) What is a Test Case?

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 scenario 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.

Test Scenario Vs Test Case

Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

For a Test Scenario: Check Login Functionality there many possible test cases are:

- Test Case 1: Check results on entering valid User Id & Password
- Test Case 2: Check results on entering Invalid User ID & Password
- Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

This is nothing but a Test Case.

The format of Standard Test Cases

Below is a format of a standard login Test cases example.

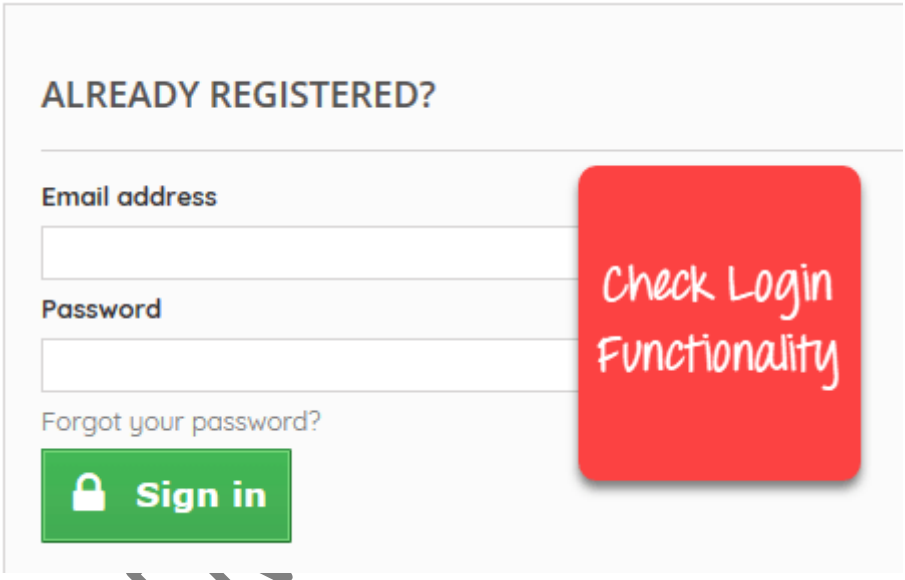
Test Case ID	Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TU01	Check Customer Login with valid Data	1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit	Userid = guru99 Password = pass99	User should Login into an application	As Expected	Pass

TU02	Check Customer Login with invalid Data	1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit	Userid = guru99 Password = glass99	User should not Login into an application	As Expected	Pass

This entire table may be created in Word, Excel or any other Test management tool. That's all to Test Case Design

How to Write Test Cases in Manual Testing

Let's create a Test Case for the scenario: Check Login Functionality



Step 1) A simple test case to explain the scenario would be

Test Case #	Test Case Description
1	Check response when valid email and password is entered

Step 2) Test the Data.

In order to execute the test case, you would need Test Data. Adding it below

Test Case #	Test Case Description	Test Data
1	Check response when valid email and password is entered	Email: guru99@email.com Password: lNf9^Oti7^2h

Identifying test data can be time-consuming and may sometimes require creating test data afresh. The reason it needs to be documented.

Step 3) Perform actions.
In order to execute a test case, a tester needs to perform a specific set of actions on the AUT. This is documented as below:

Test Case #	Test Case Description	Test Steps	Test Data
1	Check response when valid email and password is entered	1) Enter Email Address 2) Enter Password 3) Click Sign in	Email: guru99@email.com Password: lNf9^Oti7^2h

Many times the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

Step 4) Check behavior of the AUT.
The goal of test cases in software testing is to check behavior of the AUT for an expected result. This needs to be documented as below

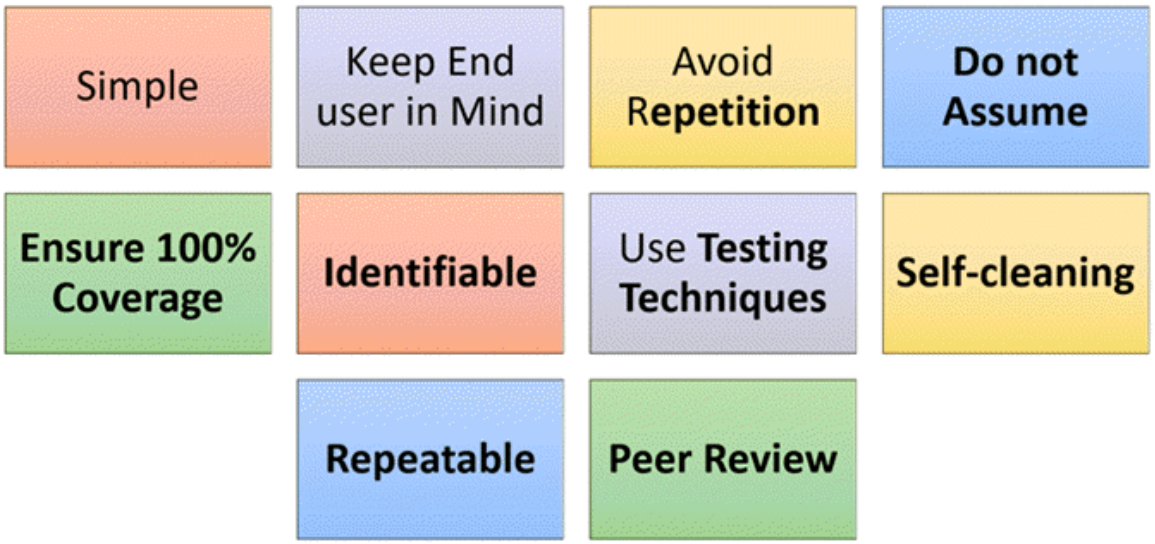
Test Case #	Test Case Description	Test Data	Expected Result
1	Check response when valid email and password is entered	Email: guru99@email.com Password: lNf9^Oti7^2h	Login should be successful

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check response when valid email and password is entered	Email: guru99@email.com Password: lNf9^Oti7^2h	Login should be successful	Login was successful	Pass

Step 5) That apart your test case -may have a field like,
Pre – Condition which specifies things that must be in place before the test can run. For our test case, a pre-condition would be to have a browser installed to have access to the site under test. A test case may also include Post – Conditions which specifies anything that applies after the test case completes. For our test case, a postcondition would be time & date of login is stored in the database

Best Practice for writing good Test Case.



Test Case Best Practice

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 the understanding the test steps easy and tests execution faster.

2. Create Test Case with End User in Mind

The ultimate goal of any software project is to create test cases that meet customer requirements and is 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 pre-condition column

4. Do not Assume

Do not assume functionality and features of your software application while preparing 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 is 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 behavior.
- **State Transition Technique**: This method is used when software behavior 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 pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

9. Repeatable and self-standing

The test case should generate the same results every time no matter who tests it

10. Peer Review.

After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.

While drafting a test case to include the following information

- The description of what requirement is being tested
- The explanation of how the system will be tested
- The test setup like a version of an application under test, software, data files, operating system, hardware, security access, physical or logical date, time of day, prerequisites such as other tests and any other setup information pertinent to the requirements being tested
- Inputs and outputs or actions and expected results
- Any proofs or attachments
- Use active case language
- Test Case should not be more than 15 steps
- An automated test script is commented with inputs, purpose and expected results

- The setup offers an alternative to pre-requisite tests
- With other tests, it should be an incorrect business scenario order

Test Case Management Tools

Test management tools are the automation tools that help to manage and maintain the Test Cases. Main Features of a test case management tool are

1. **For documenting Test Cases:** With tools, you can expedite Test Case creation with use of templates
2. **Execute the Test Case and Record the results:** Test Case can be executed through the tools and results obtained can be easily recorded.
3. **Automate the Defect Tracking:** Failed tests are automatically linked to the bug tracker, which in turn can be assigned to the developers and can be tracked by email notifications.
4. **Traceability:** Requirements, Test cases, Execution of Test cases are all interlinked through the tools, and each case can be traced to each other to check test coverage.
5. **Protecting Test Cases:** Test cases should be reusable and should be protected from being lost or corrupted due to poor version control. Test Case Management Tools offer features like

- Naming and numbering conventions
- Versioning
- Read-only storage
- Controlled access
- Off-site backup

Popular Test Management tools are: Quality Center and JIRA

E) What is Framework in Automation Testing?

A **Test Automation Framework** is a set of guidelines like coding standards, test-data handling, object repository treatment etc… which when followed during automation scripting produces beneficial outcomes like increased code re-usage, higher portability, reduced script maintenance cost etc. These are just guidelines and not rules; they are not mandatory and you can still script without following the guidelines. But you will miss out on the advantages of having a Framework.

Why do you need a Framework?

Let's consider an example to understand why you need a Framework.

I am sure you have attended a seminar/lecture/conference where the participants were asked to observe the following guidelines –

- Participants should occupy their seat 5 minutes before the start of a lecture
- Bring along a notebook and pen for note taking.

- Read the abstract so you have an idea of what the presentation will be about.
- Mobile Phones should be set on silent
- Use the exit gates at opposite end to the speaker should you require to leave in the middle of the lecture.
- Questions will be taken at the end of the session

Do you think you can conduct a seminar **WITHOUT** observing these guidelines????

The answer is a big **YES!** Certainly, you can conduct a seminar / lecture / conference / demonstration without the above guidelines (in fact some of us will not follow them even though there are laid …

But if the guidelines are followed it will result in a beneficial outcome like reduced audience distraction during lecture and increased participant retention and understanding of the subject matter.

Based on the above, a **Framework can be defined as a set of guidelines which when followed produces beneficial results.**

Types of Test Automation Frameworks

Below are the different types of Automated Testing Frameworks:

- 1) Linear Scripting
- 2) The Test Library Architecture Framework.
- 3) The Data-Driven Testing Framework.
- 4) The Keyword-Driven or Table-Driven Testing Framework.
- 5) The Hybrid Test Automation Framework.

Lets look at them in detail –

1) Linear Scripting – Record & Playback

It is the simplest of all Testing Automation Frameworks and also know as **"Record & Playback"**. In this Automation Testing Framework, Tester manually records each step (Navigation and User Inputs), Inserts Checkpoints (Validation Steps) in the first round . He then , Plays back the recorded script in the subsequent rounds.

Example: Consider logging into Flight Reservation Application and checking whether the application has loaded on successful log-on. Here , the tester will simply record the steps and add validation steps.

```
SystemUtil.Run "flight4a.exe","","","open"
Dialog("Login").WinEdit("Agent Name:").Set "Guru99"
Dialog("Login").WinEdit("Password:").Set "Mercury"
Dialog("Login").WinButton("OK").Click
'Check Flight Reservation Window has loaded after successful log-on
```

Window("Flight Reservation").Check CheckPoint("Flight Reservation")

Advantages

- Fastest way to generate a script
- Automation expertise not required
- The easiest way to learn the features of the Testing Tool

Disadvantages

- Little reuse of scripts
- Test data is hardcoded into the script
- Maintenance Nightmare

2) The Test Library Architecture Framework

It is also know as **"Structured Scripting"** or **"Functional Decomposition".**

In this Automation Testing Framework, test scripts are initially recorded by "Record & Playback" method. Later, common tasks inside the scripts are identified and grouped into Functions. These Functions are called by main test script called **Driver** in different ways to create test cases.

Example: Using the same example as above, the function for logging in to Flight Reservation will look like .

```
Function Login()
SystemUtil.Run "flight4a.exe","","","open"
Dialog("Login").WinEdit("Agent Name:").Set "Guru99"
Dialog("Login").WinEdit("Password:").Set "Mercury"
Dialog("Login").WinButton("OK").Click
End Function
```

Now, you will call this function in the main script as follows

```
Call Login()
---------------------------
Other Function calls / Test Steps.
---------------------------
```

Advantages

- Higher level of code reuse is achieved in Structured Scripting as compared to "Record & Playback"
- The automation scripts are less costly to develop due to higher code re-use
- Easier Script Maintenance

Disadvantages

- Technical expertise is necessary to write Scripts using Test Library Framework
- More time is needed to plan and prepare test scripts.
- Test Data is hard coded within the scripts

3) The Data-Driven Testing Framework

In this Framework , while Test Case logic resides in Test Scripts, the Test Data is separated and kept outside the Test Scripts. Test Data is read from the external files (Excel Files, Text Files, CSV Files, ODBC Sources, DAO Objects, ADO Objects) and are loaded into the variables inside the Test Script. Variables are used both for Input values and for Verification values. Test Scripts themselves are prepared either using Linear Scripting or Test Library Framework.

Example: Developing the Flight Reservation Login script using this method will involve two steps.

Step 1) Create a Test – Data file which could be Excel , CSV , or any other database source.

AgentName	Password
Jimmy	Mercury
Tina	MERCURY
Bill	MerCURY

Step 2) Develop Test Script and make references to your Test- Data source.

```
SystemUtil.Run "flight4a.exe","","","open"
Dialog("Login").WinEdit("Agent Name:").Set DataTable("AgentName", dtGlobalSheet)
Dialog("Login").WinEdit("Password:").Set DataTable("Password", dtGlobalSheet)
Dialog("Login").WinButton("OK").Click
'Check Flight Reservation Window has loaded
Window("Flight Reservation").Check CheckPoint("Flight Reservation")
```

**Note "dtGlobalSheet" is the default excel sheet provided by QTP.

Advantages

- Changes to the Test Scripts do not affect the Test Data
- Test Cases can be executed with multiple Sets of Data
- A Variety of Test Scenarios can be executed by just varying the Test Data in the External Data File

Disadvantages

- More time is needed to plan and prepare both Test Scripts and Test Data

4) The Keyword-Driven or Table-Driven Testing Framework

The Keyword-Driven or Table-Driven automation framework development requires data tables and keywords, **independent of the test automation tool** used to execute them . Tests can be designed with or without the Application. In a keyword-driven test, the functionality of the application-under-test is documented in a table as well as in step-by-step instructions for each test.

There are 3 basic components of a Keyword Driven Framework viz. Keyword , Application Map , Component Function.

What is a Keyword?

Keyword is an Action that can be performed on a GUI Component. Ex. For GUI Component Textbox some Keywords (Action) would be InputText, VerifyValue, VerifyProperty and so on.

What is the Application Map?

An Application Map Provides Named References for GUI Components. Application Maps are nothing but "**Object Repository**"

What is Component Function?

Component Functions are those functions that actively manipulate or interrogate the GUI component. An example of a function would be click on web button with all error handling , enter data in a Web Edit with all error handling. Component functions could be application dependent or independent.

Example: To understand Keyword View lets take the same example. It invovles 2 steps

Step 1: Creating Data Table (Different from Test-Data Table created in Data Driven Framework). This Data Table contains Action to be performed on GUI Objects and corresponding arguments if any. Each row respresents one Test Step.

Object	**Action**	
(Application MAP)	**(KEYWORDS)**	**Argument**
WinEdit(Agent Name)	Set	Guru99
WinEdit(Password)	Set	Mercury
WinButton(OK)	Click	
Window(Flight Reservation)	Verify	Exists

Step 2: Writing Code in the form of Component Functions.

Once you've created your data table(s), you simply write a program or a set of scripts that reads in each step, executes the step based on the keyword contained the Action field, performs error checking, and logs any relevant information. This program or set of scripts would look similar to the pseudo code below:

```
Function main()
{
  Call ConnectTable(Name of the Table) { //Calling Function for connecting to the table.
    while (Call TableParser() != -1) //Calling function for Parsing and extracting values from the table.
    {
      Pass values to appropriate COMPONENT functions.Like Set(Object Name, Argument) ex.Set(Agent Name, Guru99).
    }
  }
  Call CloseConnection() //Function for Closing connection after all the operation has been performed.
} //End of main
```

Thats all to Keyword Driven Framework.

The advantage of Keyword Driven Framework is that the Keywords are re-usable. To understand this consider you want to verify login operation for a Website say YAHOO MAIL. The table will look like this –

Object	**Action**	
(APPLICATION MAP)	**(KEYWORD)**	**Argument**
WebEdit(UserName)	Set	abc@yahoo.com
WebEdit(Password)	Set	xxxxx
WebButton(OK)	Click	
Window(Yahoo Mail)	Verify	Loads

If you observe in this case the Keywords Set , Click , Verify remain the same for which corresponding component functions are already developed. All you need to do is change the Application Mapping (Object Repository) from earlier Flight Reservation to Yahoo Mail , with a change in argument values and the same script will work!

Advantages

- Provides high code re-usability
- Test Tool Independent
- Independent of Application Under Test, the same script works for AUT (with some limitations)
- Tests can be designed with or without AUT

Disadvantages

- Initial investment being pretty high, the benefits of this can only be realized if the application is considerably big and the test scripts are to be maintained for quite a few years.
- High Automation expertise is required to create the Keyword Driven Framework.

NOTE : Even though Micro Focus UFT advertises itself as KeyWord Driven Framework, you can not achieve complete test tool and application idependence using HP UFT.

5) The Hybrid Test Automation Framework

As the name suggests this framework is the combination of one or more Automation Frameworks discussed above pulling from their strengths and trying to mitigate their weaknesses. The hybrid test QA automation framework is what most test automation frameworks evolve into over time and multiple projects. Maximum industry uses Keyword Framework in a combination of Function decomposition method.

PS: Other Automation Frameworks worth a mention are

Test Modularity Framework

In this framework, a common task in test script are grouped together as Modules.

Example: Using Actions in QTP use can create a Modualr Scripts

Sample Script for Login

```
SystemUtil.Run "flight4a.exe","","","open"
Dialog("Login").WinEdit("Agent Name:").Set "Guru99"
Dialog("Login").WinEdit("Password:").Set "Mercury"
Dialog("Login").WinButton("OK").Click
'End of Script
```

Now you can call this Action in the main script as follows –

RunAction ("Login[Argument]", oneIteration)

Business Process Testing (BPT)

These Automation Frameworks, break up large Business Processes into Components which can re-used multiple times in the same or different test scripts. For example , the Business Process of Booking a flight is split into components like Login , Finding Flights , Booking , Payment & Logout which can be re-used in the same Business process or different processes. Also, BPT facilitates closer coordination amongst SME's and Automation Engineers .

Benefits of Test Automation Framework Architecture

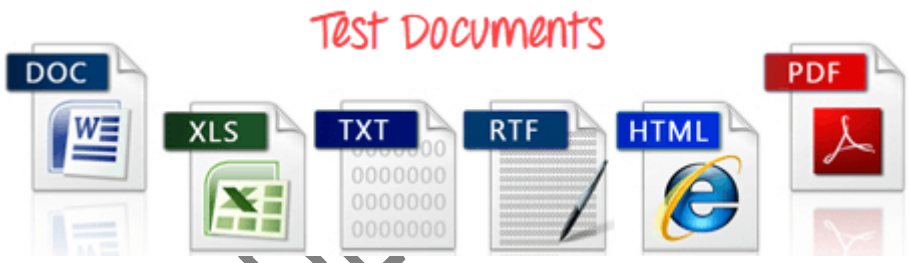
Following are the benefits of Test automation framework architecture:

- A test automation framework helps to reduce the risk and cost expenses
- It improves the efficiency of tests
- It helps to reduce the maintenance cost
- Allows the reuse of code
- It allows achieving maximum test coverage
- It maximizes the application functionality
- Helps to reduce test case duplication
- It helps to improve the test efficiency and performance with test automation

F)What is Test Documentation?

Test documentation is documentation of artifacts created before or during the testing of software. It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc. It is a complete suite of documents that allows you to describe and document test planning, test design, test execution, test results that are drawn from the testing activity.

Why Test Formality?



For a newbie, it's easy to assume that Testing is executing the various section of code on an ad-hoc basis and verifying the results. But in the real world, Testing is a very formal activity and is documented in detail. Test Documentation makes planning, review, and execution of testing easy as well as verifiable.

The degree of test formality depends on

- The type of application under test
- Standards followed by your organization
- The maturity of the development process.

Testing activities generally consume 30% to 50% of software development project effort. Documentations help to identify Test process improvement that can be applied to future projects.

Examples of Test Documentation

Here, are important Types of Test Documentation:

Types of Testing Documents	Description

Test policy	It is a high-level document which describes principles, methods and all the important testing goals of the organization.
Test strategy	A high-level document which identifies the Test Levels (types) to be executed for the project.
Test plan	A test plan is a complete planning document which contains the scope, approach, resources, schedule, etc. of testing activities.
Requirements Traceability Matrix	This is a document which connects the requirements to the test cases.
Test Scenario	Test scenario is an item or event of a software system which could be verified by one or more Test cases.
Test case	It is a group of input values, execution preconditions, expected execution postconditions and results. It is developed for a Test Scenario.
Test Data	Test Data is a data which exists before a test is executed. It used to execute the test case.
Defect Report	Defect report is a documented report of any flaw in a Software System which fails to perform its expected function.
Test summary report	Test summary report is a high-level document which summarizes testing activities conducted as well as the test result.

Best practice to Achieve Test Documentation

- QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
- Don't just create and leave the document, but update whenever required
- Use version control to manage and track your documents
- Try to document what is needed for you to understand your work and what you will need to produce to your stakeholders
- You should use a standard template for documentation like excel sheet or doc file
- Store all your project related documents at a single location. It should be accessible to every team member for reference as well as to update when needed
- Not providing enough detail is also a common mistake while creating a test document

Advantages of Test Documentation

- The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities. Helps you to remove ambiguity which often arises when it comes to the allocation of tasks
- Documentation not only offers a systematic approach to software testing, but it also acts as training material to freshers in the software testing process
- It is also a good marketing & sales strategy to showcase Test Documentation to exhibit a mature testing process
- Test documentation helps you to offer a quality product to the client within specific time limits
- In Software Engineering, Test Documentation also helps to configure or set-up the program through the configuration document and operator manuals
- Test documentation helps you to improve transparency with the client

Disadvantages of Test Documentation

- The cost of the documentation may surpass its value as it is very time-consuming
- Many times, it is written by people who can't write well or who don't know the material
- Keeping track of changes requested by the client and updating corresponding documents is tiring.
- Poor documentation directly reflects the quality of the product as a misunderstanding between the client and the organization can occur

A)What is Static Testing?

Static Testing is a software testing technique which is used to check defects in software application without executing the code. Static testing is done to avoid errors at an early stage of development as it is easier to identify the errors and solve the errors. It also helps finding errors that may not be found by Dynamic Testing.
Its counterpart is Dynamic Testing which checks an application when the code is run. Refer to this tutorial for a detailed difference between static and dynamic testing.

The two main types of static testing techniques are

- **Manual examinations**: Manual examinations include analysis of code done manually, also known as **REVIEWS.**
- **Automated analysis using tools:** Automated analysis are basically static analysis which is done using tools.

What is Testing Review?

A review in a Static Testing is a process or meeting conducted to find the potential defects in the design of any program. Another significance of review is that all the team members get to know about the progress of the project and sometimes the diversity of thoughts may result in excellent suggestions. Documents are directly examined by people and discrepancies are sorted out.

Reviews can further be classified into four parts:

- Informal reviews
- Walkthroughs
- Technical review
- Inspections

During the Review process four types of participants that take part in testing are:

- **Moderator**: Performs entry check, follow up on rework, coaching team member, schedule the meeting.
- **Author**: Takes responsibility for fixing the defect found and improves the quality of the document
- **Scribe**: It does the logging of the defect during a review and attends the review meeting
- **Reviewer**: Check material for defects and inspects
- **Manager**: Decide on the execution of reviews and ensures the review process objectives are met.

Types of defects which can be easier to find during static testing are:

- Deviations from standards
- Non-maintainable code
- Design defects
- Missing requirements
- Inconsistent interface specifications

Usually, the defect discovered during static testing are due to security vulnerabilities, undeclared variables, boundary violations, syntax violations, inconsistent interface, etc.

Why Static Testing?

Static testing is performed due to the following reasons

- Early defect detection and correction
- Reduced development timescales
- Reduced testing cost and time
- For improvement of development productivity
- To get fewer defect at a later stage of testing

What is Tested in Static Testing

In Static Testing, following things are tested

- Unit Test Cases
- Business Requirements Document (BRD)
- Use Cases
- System/Functional Requirements
- Prototype
- Prototype Specification Document
- DB Fields Dictionary Spreadsheet
- Test Data
- Traceability Matrix Document
- User Manual/Training Guides/Documentation
- Test Plan Strategy Document/Test Cases
- Automation/Performance Test Scripts

How Static Testing is Performed

To perform Static Testing, it is done in the following ways,

- Carry out the inspection process to completely inspect the design of the application
- Use a checklist for each document under review to ensure all reviews are covered completely

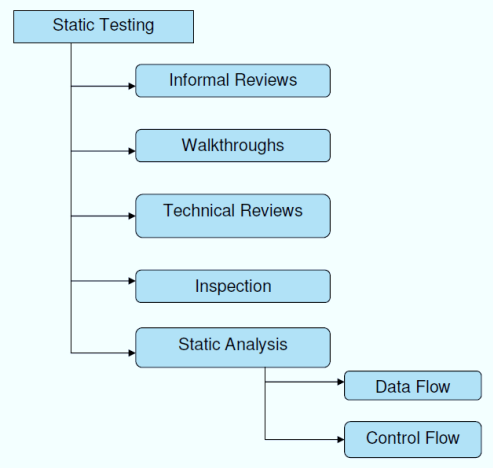
The various activities for performing Static Testing are:

1. **Use Cases Requirements Validation:** It validates that all the end-user actions are identified, as well as any input and output associated with them. The more detailed and thorough the use cases are, the more accurate and comprehensive the test cases can be.
2. **Functional Requirements Validation**: It ensures that the Functional Requirements identify all necessary elements. It also looks at the database functionality, interface listings, and hardware, software, and network requirements.
3. **Architecture Review**: All business level process like server locations, network diagrams, protocol definitions, load balancing, database accessibility, test equipment, etc.
4. **Prototype/Screen Mockup Validation**: This stage includes validation of requirements and use cases.

5. **Field Dictionary Validation**: Every field in the UI is defined well enough to create field level validation test cases. Fields are check for min/max length, list values, error messages, etc.

Static Testing Techniques

- Informal Reviews
- Walkthroughs
- Technical Reviews
- Inspections
- Static Analysis
    - Data Flow
    - Control Flow



B) Data Flow Testing

Data Flow Testing is a type of structural testing. It is a method that is used to find the test paths of a program according to the locations of definitions and uses of variables in the program. It has nothing to do with data flow diagrams.
It is concerned with:

- Statements where variables receive values,
- Statements where these values are used or referenced.

To illustrate the approach of data flow testing, assume that each statement in the program assigned a unique statement number. For a statement number S-

DEF(S) = {X | statement S contains the definition of X}

USE(S) = {X | statement S contains the use of X}

If a statement is a loop or if condition then its DEF set is empty and USE set is based on the condition of statement s.

Data Flow Testing uses the control flow graph to find the situations that can interrupt the flow of the program.
Reference or define anomalies in the flow of the data are detected at the time of associations between values and variables. These anomalies are:

- A variable is defined but not used or referenced,
- A variable is used but never defined,
- A variable is defined twice before it is used

Advantages of Data Flow Testing:
Data Flow Testing is used to find the following issues-

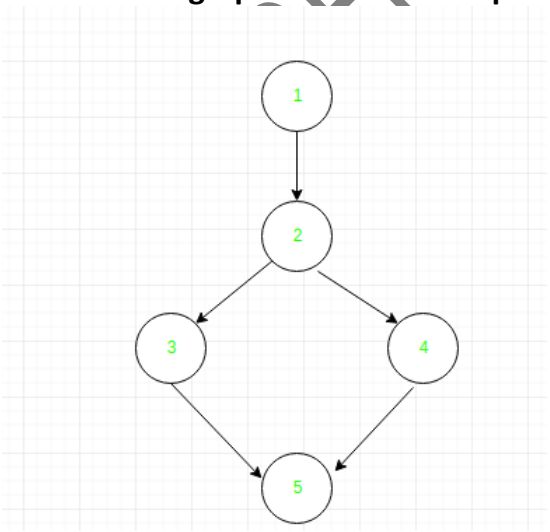
- To find a variable that is used but never defined,
- To find a variable that is defined but never used,
- To find a variable that is defined multiple times before it is use,
- Deallocating a variable before it is used.

Disadvantages of Data Flow Testing

- Time consuming and costly process
- Requires knowledge of programming languages

Example:
1. read x, y;
2. if(x>y)
3. a = x+1
else
4. a = y-1
5. print a;

Control flow graph of above example:



Define/use of variables of above example:

Variable	Defined at node	Used at node
x	1	2, 3

y	1	2, 4
a	3, 4	5

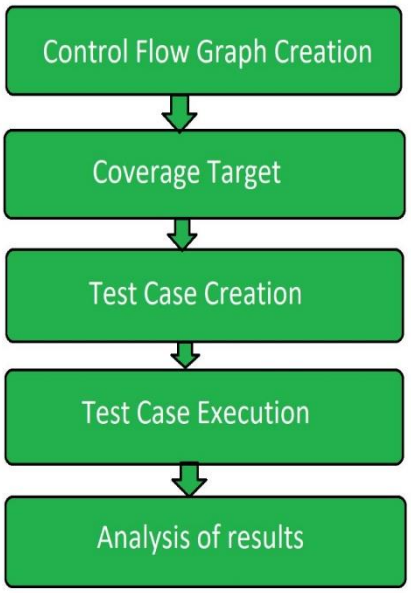
C) Control Flow Software Testing

Control flow testing is a type of software testing that uses program's control flow as a model. Control flow testing is a structural testing strategy. This testing technique comes under white box testing. For the type of control flow testing, all the structure, design, code and implementation of the software should be known to the testing team.

This type of testing method is often used by developers to test their own code and own implementation as the design, code and the implementation is better known to the developers. This testing method is implemented with the intention to test the logic of the code so that the user requirements can be fulfilled. Its main application is to relate the small programs and segments of the larger programs.

Control Flow Testing Process:

Following are the steps involved into the process of control flow testing:



- **Control Flow Graph Creation:**
  From the given source code a control flow graph is created either manually or by using the software.
- **Coverage Target:**
  A coverage target is defined over the control flow graph that includes nodes, edges, paths, branches etc.

- **Test Case Creation:**
  Test cases are created using control flow graphs to cover the defined coverage target.
- **Test Case Execution:**
  After the creation of test cases over coverage target, further test cases are executed.
- **Analysis:**
  Analyze the result and find out whether the program is error free or has some defects.

Control Flow Graph**:**
Control Flow Graph is a graphical representation of control flow or computation that is done during the execution of the program. Control flow graphs are mostly used in static analysis as well as compiler applications, as they can accurately represent the flow inside of a program unit. Control flow graph was originally developed by Frances E. Allen.

Advantages of Control flow testing:

- It detects almost half of the defects that are determined during the unit testing.
- It also determines almost one-third of the defects of the whole program.
- It can be performed manually or automated as the control flow graph that is used can be made by hand or by using software also.

Disadvantages of Control flow testing:

- It is difficult to find missing paths if program and the model are done by same person.
- Unlikely to find spurious features.

D) Cyclomatic Complexity

Cyclomatic complexity of a code section is the quantitative measure of the number of linearly independent paths in it. It is a software metric used to indicate the complexity of a program. It is computed using the Control Flow Graph of the program. The nodes in the graph indicate the smallest group of commands of a program, and a directed edge in it connects the two nodes i.e. if second command might immediately follow the first command.

For example, if source code contains no control flow statement then its cyclomatic complexity will be 1 and source code contains a single path in it. Similarly, if the source code contains one **if condition** then cyclomatic complexity will be 2 because there will be two paths one for true and the other for false.
Mathematically, for a structured program, the directed graph inside control flow is the edge joining two basic blocks of the program as control may pass from first to

second.
So, cyclomatic complexity M would be defined as,

M = E – N + 2P

where,
E = the number of edges in the control flow graph
N = the number of nodes in the control flow graph
P = the number of connected components

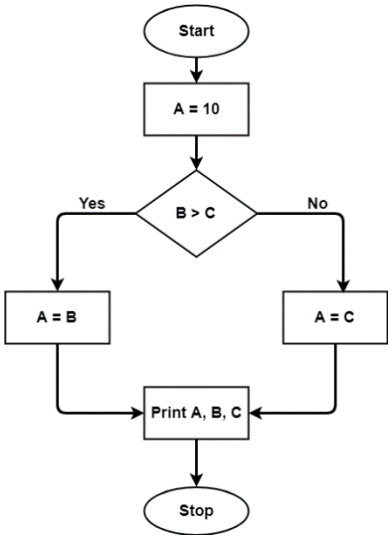
Steps that should be followed in calculating cyclomatic complexity and test cases design are:

- Construction of graph with nodes and edges from code.
- Identification of independent paths.
- Cyclomatic Complexity Calculation
- Design of Test Cases

Let a section of code as such:

```
A = 10
  IF B > C THEN
    A = B
  ELSE
    A = C
  ENDIF
Print A
Print B
Print C
```

Control Flow Graph of above code



The cyclomatic complexity calculated for above code will be from control flow graph. The graph shows seven shapes(nodes), seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

Use of Cyclomatic Complexity:

- Determining the independent path executions thus proven to be very helpful for Developers and Testers.
- It can make sure that every path have been tested at least once.
- Thus help to focus more on uncovered paths.
- Code coverage can be improved.
- Risk associated with program can be evaluated.
- These metrics being used earlier in the program helps in reducing the risks.

Advantages of Cyclomatic Complexity:.

- It can be used as a quality metric, gives relative complexity of various designs.
- It is able to compute faster than the Halstead's metrics.
- It is used to measure the minimum effort and best areas of concentration for testing.
- It is able to guide the testing process.
- It is easy to apply.

Disadvantages of Cyclomatic Complexity:

- It is the measure of the programs's control complexity and not the data complexity.
- In this, nested conditional structures are harder to understand than non-nested structures.
- In case of simple comparisons and decision structures, it may give a misleading figure.

E) White Box Testing

White Box Testing is a testing technique in which software's internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, and Glass box testing.
It is one of two parts of the Box Testing approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user perspective. On the other hand, White box testing in software engineering is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner

workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration, and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

We have divided it into two basic steps to give you a simplified explanation of white box testing. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

STEP 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

White Box Testing Techniques

A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product

There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques a box tester can use:

Statement Coverage:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

Branch Coverage – This technique checks every possible path (if-else and other conditional loops) of a software application.

Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. **Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.**

Following are important WhiteBox Testing Techniques:

- Statement Coverage
- Decision Coverage
- Branch Coverage
- Condition Coverage
- Multiple Condition Coverage
- Finite State Machine Coverage
- Path Coverage
- Control flow testing
- Data flow testing

Advantages of White Box Testing

- Code optimization by finding hidden errors.
- White box tests cases can be easily automated.
- Testing is more thorough as all code paths are usually covered.
- Testing can start early in SDLC even if GUI is not available.

Disadvantages of WhiteBox Testing

- White box testing can be quite complex and expensive.
- Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed and can lead to production errors.
- White box testing requires professional resources with a detailed understanding of programming and implementation.
- White-box testing is time-consuming, bigger programming applications take the time to test fully.

F)Statement Coverage Testing

Statement coverage is one of the widely used software testing. It comes under white box testing.

Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code out of total statements present in the source code.

Statement coverage derives scenario of test cases under the white box testing process which is based upon the structure of the code.

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

In white box testing, concentration of the tester is on the working of internal source code and flow chart or flow graph of the code.

Generally, in the internal source code, there is a wide variety of elements like operators, methods, arrays, looping, control statements, exception handlers, etc. Based on the input given to the program, some code statements are executed and some may not be executed. The goal of statement coverage technique is to cover all the possible executing statements and path lines in the code.

Let's understand the process of calculating statement coverage by an example:

Here, we are taking source code to create two different scenarios according to input values to check the percentage of statement coverage for each scenario.

Source Code Structure:

- Take input of two values like a=0 and b=1.
- Find the sum of these two values.
- If the sum is greater than 0, then print "This is the positive result."
- If the sum is less than 0, then print "This is the negative result."

```
1. input (int a, int b)
2. {
3. Function to print sum of these integer values (sum = a+b)
4. If (sum>0)
5. {
6. Print (This is positive result)
7. } else
8. {
9. Print (This is negative result)
10. }
11. }
```

So, this is the basic structure of the program, and that is the task it is going to do.

Now, let's see the two different scenarios and calculation of the percentage of Statement Coverage for given source code.

Scenario 1:
If a = 5, b = 4

```
1. print (int a, int b) {
2. int sum = a+b;
3. if (sum>0)
4. print ("This is a positive result")
5. else
6. print ("This is negative result")
7. }
```

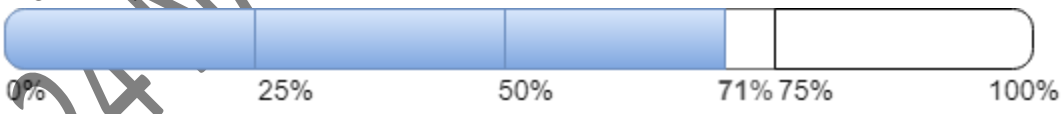
In scenario 1, we can see the value of sum will be 9 that is greater than 0 and as per the condition result will be "**This is a positive result.**" The statements highlighted in yellow color are executed statements of this scenario.

To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 5.

1. Total number of statements = 7
2. Number of executed statements = 5

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

1. Statement coverage = 5/7*100
2. = 500/7
3. = 71%



Likewise, in scenario 2,

Scenario 2:
If A = -2, B = -7

```
1. print (int a, int b) {
2. int sum = a+b;
3. if (sum>0)
4. print ("This is a positive result")
```

5. **else**
6. print ("This is negative result")
7. }

In scenario 2, we can see the value of sum will be -9 that is less than 0 and as per the condition, result will be "**This is a negative result.**" The statements highlighted in yellow color are executed statements of this scenario.

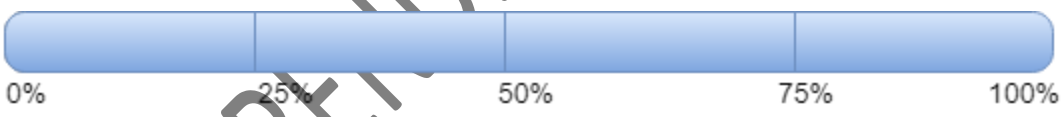
To calculate statement coverage of the first scenario, take the total number of statements that is 7 and the number of used statements that is 6.

Total number of statements = 7
Number of executed statements = 6

$$\text{Statement coverage} = \frac{\text{Number of executed statements}}{\text{Total number of statements}} * 100$$

1. Statement coverage = 6/7*100
2. = 600/7
3. = 85%

But, we can see all the statements are covered in both scenario and we can consider that the overall statement coverage is 100%.



So, the statement coverage technique covers dead code, unused code, and branches.

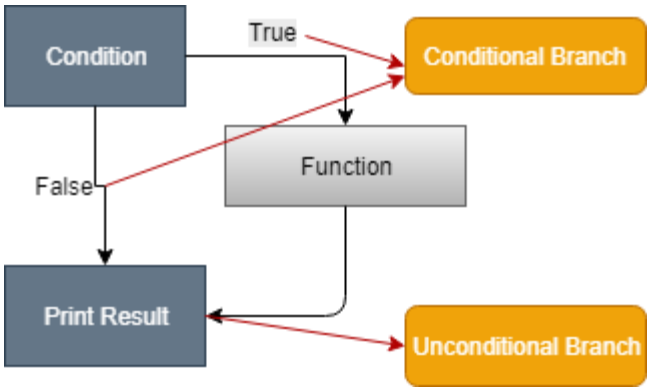
G) Branch Coverage Testing

Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once. Branch coverage technique is a whitebox testing technique that ensures that every branch of each decision point must be executed.

However, branch coverage technique and decision coverage technique are very similar, but there is a key difference between the two. Decision coverage technique covers all branches of each decision point whereas branch testing covers all branches of every decision point of the code.

In other words, branch coverage follows decision point and branch coverage edges. Many different metrics can be used to find branch coverage and decision coverage, but some of the most basic metrics are: finding the percentage of program and paths of execution during the execution of the program.

Like decision coverage, it also uses a control flow graph to calculate the number of branches.



How to calculate Branch coverage?

There are several methods to calculate Branch coverage, but pathfinding is the most common method.

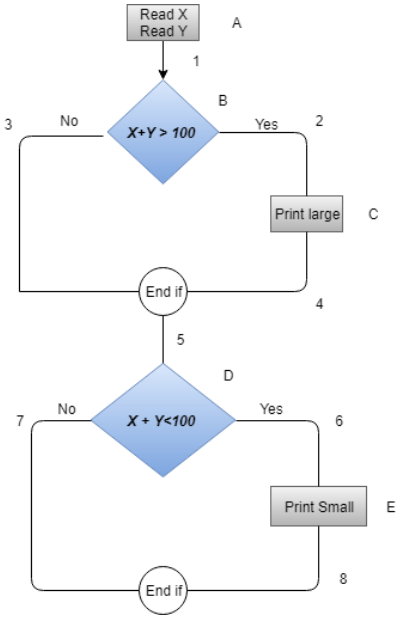
In this method, the number of paths of executed branches is used to calculate Branch coverage. Branch coverage technique can be used as the alternative of decision coverage. Somewhere, it is not defined as an individual technique, but it is distinct from decision coverage and essential to test all branches of the control flow graph.

Let's understand it with an example:

1. Read X
2. Read Y
3. IF X+Y > 100 THEN
4. Print "Large"
5. ENDIF
6. If X + Y<100 THEN
7. Print "Small"
8. ENDIF

This is the basic code structure where we took two variables X and Y and two conditions. If the first condition is true, then print "Large" and if it is false, then go to the next condition. If the second condition is true, then print "Small."

Control flow graph of code structure



In the above diagram, control flow graph of code is depicted. In the first case traversing through "Yes "decision, the path is **A1-B2-C4-D6-E8**, and the number of covered edges is 1, 2, 4, 5, 6 and 8 but edges 3 and 7 are not covered in this path. To cover these edges, we have to traverse through "No" decision. In the case of "No" decision the path is A1-B3-5-D7, and the number of covered edges is 3 and 7. So by traveling through these two paths, all branches have covered.

Path 1 - A1-B2-C4-D6-E8

Path 2 - A1-B3-5-D7

Branch Coverage (BC) = Number of paths = 2

Case	Covered Branches	Path	Branch coverage
Yes	1, 2, 4, 5, 6, 8	A1-B2-C4-D6-E8	2
No	3,7	A1-B3-5-D7	

H) Path Testing:

What is Path Testing?

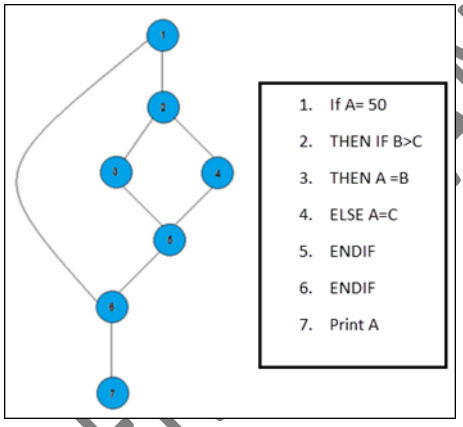
Path testing is a structural testing method that involves using the source code of a program in order to find every possible executable path. It helps to determine all faults lying within a piece of code. This method is designed to execute all or selected path through a computer program.

Any software program includes, multiple entry and exit points. Testing each of these points is a challenging as well as time-consuming. In order to reduce the redundant tests and to achieve maximum test coverage, basis path testing is used.

Basis Path Testing in Software Engineering

Basis Path Testing in software engineering is a White Box Testing method in which test cases are defined based on flows or logical paths that can be taken through the program. The objective of basis path testing is to define the number of independent paths, so the number of test cases needed can be defined explicitly to maximize test coverage.
In software engineering, Basis path testing involves execution of all possible blocks in a program and achieves maximum path coverage with the least number of test cases. It is a hybrid method of branch testing and path testing methods.

Here we will take a simple example, to get a better idea what is basis path testing include



In the above example, we can see there are few conditional statements that is executed depending on what condition it suffice. Here there are 3 paths or condition that need to be tested to get the output,

- **Path 1**: 1,2,3,5,6, 7
- **Path 2**: 1,2,4,5,6, 7
- **Path 3**: 1, 6, 7

Steps for Basis Path testing

The basic steps involved in basis path testing include

- Draw a control graph (to determine different program paths)
- Calculate Cyclomatic complexity (metrics to determine the number of independent paths)
- Find a basis set of paths
- Generate test cases to exercise each path

Advantages of Basic Path Testing

- It helps to reduce the redundant tests

- It focuses attention on program logic
- It helps facilitates analytical versus arbitrary case design
- Test cases which exercise basis set will execute every statement in a program at least once

I) State Transition Testing

State Transition Testing is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test(AUT). State transition testing helps to analyze behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behavior.
It is the model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before, is a finite state system.

State Transition Testing Technique is helpful where you need to **test different system transitions.**

When to Use State Transition?

- This can be used when a tester is testing the application for a finite set of input values.
- When the tester is trying to test sequence of events that occur in the application under test. I.e., this will allow the tester to test the application behavior for a sequence of input values.
- When the system under test has a dependency on the events/values in the past.

When to Not Rely On State Transition?

- When the testing is not done for sequential input combinations.
- If the testing is to be done for different functionalities like exploratory testing

Four Parts Of State Transition Diagram

There are 4 main components of the State Transition Model as below

1) States that the software might get

2) Transition from one state to another

3) Events that origin a transition like closing a file or withdrawing money

4) Actions that result from a transition (an error message or being given the cash.)

State Transition Diagram and State Transition Table

There are two main ways to represent or design state transition, State transition diagram, and state transition table.

In state transition diagram the states are shown in boxed texts, and the transition is represented by arrows. It is also called State Chart or Graph. It is useful in identifying valid transitions.

In state transition table all the states are listed on the left side, and the events are described on the top. Each cell in the table represents the state of the system after the event has occurred. It is also called State Table. It is useful in identifying invalid transitions.

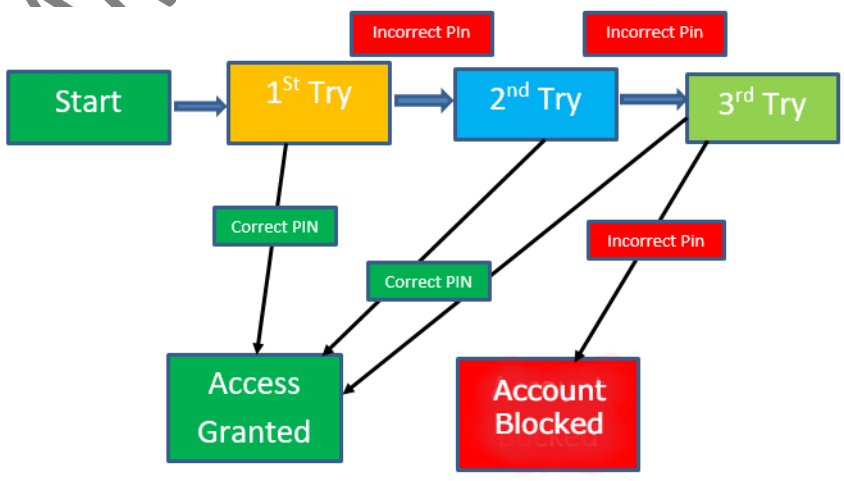
How to Make a State Transition

Example :

Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

State transition diagram



In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

State Transition Table

	Correct PIN	**Incorrect PIN**
S1) Start	**S5**	**S2**
S2) 1st attempt	**S5**	**S3**
S3) 2nd attempt	**S5**	**S4**
S4) 3rd attempt	**S5**	**S6**
S5) Access Granted	**–**	**–**
S6) Account blocked	**–**	**–**

In the table when the user enters the correct PIN, state is transitioned to S5 which is Access granted. And if the user enters a wrong password he is moved to next state. If he does the same 3rd time, he will reach the account blocked state.

Advantages and Disadvantages of State Transition Technique

Advantages	**Disadvantages**
This testing technique will provide a pictorial or tabular representation of system behavior which will make the tester to cover and understand the system behavior effectively.	The main disadvantage of this testing technique is that we can't rely in this technique every time. For example, if the system is not a finite system (not in sequential order), this technique cannot be used.
By using this testing, technique tester can verify that all the conditions are covered, and the results are captured	Another disadvantage is that you have to define all the possible states of a system. While this is all right for small systems, it soon breaks down into larger systems as there is an exponential progression in the number of states.

A) Black Box Testing

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing.

The above Black-Box can be any software system you want to test. For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

How to do BlackBox Testing

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones –

- **Functional testing** – This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** – This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** – Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

- For Functional/ Regression Tests you can use – QTP, Selenium
- For Non-Functional Tests, you can use – LoadRunner, Jmeter

Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

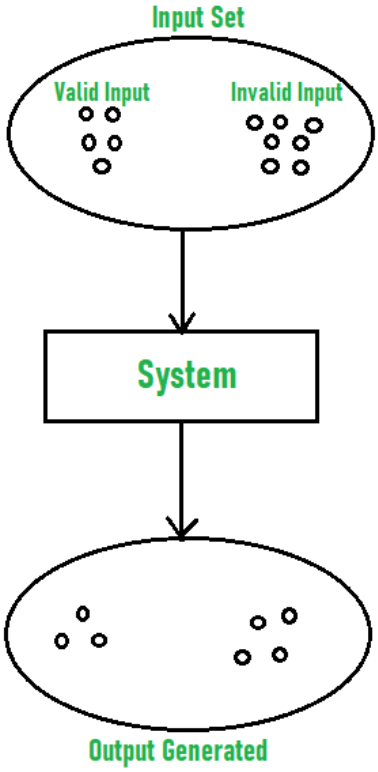
B) Equivalence Partitioning Method

Equivalence Partitioning Method is also known as Equivalence class partitioning (ECP). It is a software testing technique or black-box testing that divides input domain into classes of data, and with the help of these classes of data, test cases can be derived. An ideal test case identifies class of error that might require many arbitrary test cases to be executed before general error is observed.

In equivalence partitioning, equivalence classes are evaluated for given input conditions. Whenever any input is given, then type of input condition is checked, then for this input conditions, Equivalence class represents or describes set of valid or invalid states.

Guidelines for Equivalence Partitioning :

- If the range condition is given as an input, then one valid and two invalid equivalence classes are defined.
- If a specific value is given as input, then one valid and two invalid equivalence classes are defined.
- If a member of set is given as an input, then one valid and one invalid equivalence class is defined.
- If Boolean no. is given as an input condition, then one valid and one invalid equivalence class is defined.



Example:
Let us consider an example of any college admission process. There is a college that gives admissions to students based upon their percentage.
Consider percentage field that will accept percentage only between 50 to 90 %, more and even less than not be accepted, and application will redirect user to an error page. If percentage entered by user is less than 50 %or more than 90 %, that equivalence partitioning method will show an invalid percentage. If percentage entered is between 50 to 90 %, then equivalence partitioning method will show valid percentage.

Equivalence Partitioning		
Invalid	Valid	Invalid
<=50	50-90	>=90

C) Boundary Value Analysis

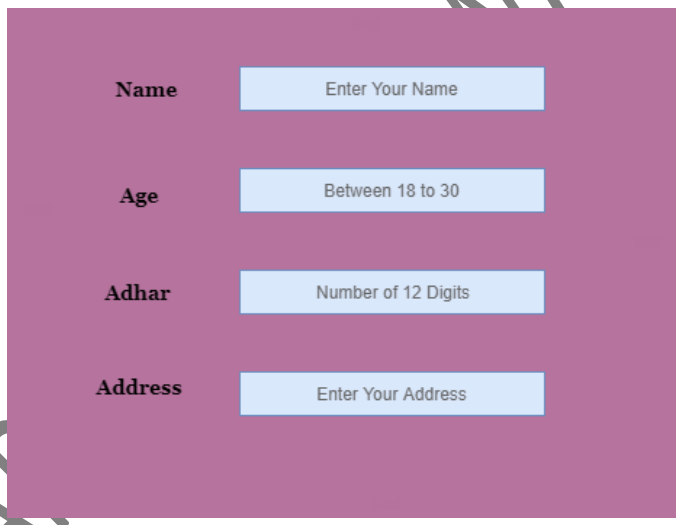
Boundary value analysis is one of the widely used case design technique for black box testing. It is used to test boundary values because the input values near the boundary have higher chances of error.

Whenever we do the testing by boundary value analysis, the tester focuses on, while entering boundary value whether the software is producing correct output or not.

Boundary values are those that contain the upper and lower limit of a variable. Assume that, age is a variable of any function, and its minimum value is 18 and the maximum value is 30, both 18 and 30 will be considered as boundary values.

The basic assumption of boundary value analysis is, the test cases that are created using boundary values are most likely to cause an error.

There is 18 and 30 are the boundary values that's why tester pays more attention to these values, but this doesn't mean that the middle values like 19, 20, 21, 27, 29 are ignored. Test cases are developed for each and every value of the range.



Testing of boundary values is done by making valid and invalid partitions. Invalid partitions are tested because testing of output in adverse condition is also essential.

Let's understand via practical:

Imagine, there is a function that accepts a number between 18 to 30, where 18 is the minimum and 30 is the maximum value of valid partition, the other values of this partition are 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 and 29. The invalid partition consists of the numbers which are less than 18 such as 12, 14, 15, 16 and 17, and more than 30 such as 31, 32, 34, 36 and 40. Tester develops test cases for both valid and invalid partitions to capture the behavior of the system on different input conditions.

Invalid test cases	Valid test cases	Invalid test cases
11, 13, 14, 15, 16, 17	18, 19, 24, 27, 28, 30	31, 32, 36, 37, 38, 39

The software system will be passed in the test if it accepts a valid number and gives the desired output, if it is not, then it is unsuccessful. In another scenario, the software system should not accept invalid numbers, and if the entered number is invalid, then it should display error massage.

If the software which is under test, follows all the testing guidelines and specifications then it is sent to the releasing team otherwise to the development team to fix the defects.

D) **Cause and Effect Graph**

The black box testing technique which underlines the relationship between a given result and all the factors affecting the result. It is used to write dynamic test cases.

The dynamic test cases are used when code works dynamically based on user input. For example, while using email account, on entering valid email, the system accepts it but, when you enter invalid email, it throws an error message. In this technique, the input conditions are assigned with causes and the result of these input conditions with effects.

Cause-Effect graph technique is based on a collection of requirements and used to determine minimum possible test cases which can cover a maximum test area of the software.

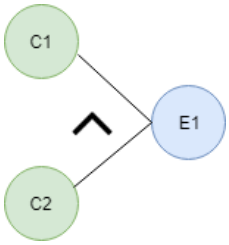
The main advantage of cause-effect graph testing is, it reduces the time of test execution and cost.

This technique aims to reduce the number of test cases but still covers all necessary test cases with maximum coverage to achieve the desired application quality.

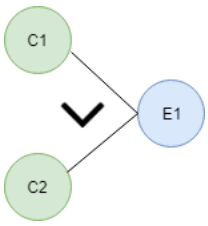
Cause-Effect graph technique converts the requirements specification into a logical relationship between the input and output conditions by using logical operators like AND, OR and NOT.

Notations used in the Cause-Effect Graph

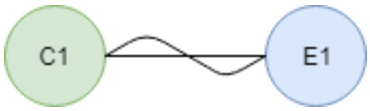
AND - E1 is an effect and C1 and C2 are the causes. If both C1 and C2 are true, then effect E1 will be true.



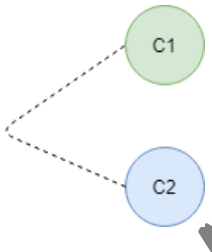
OR - If any cause from C1 and C2 is true, then effect E1 will be true.



NOT - If cause C1 is false, then effect E1 will be true.



Mutually Exclusive - When only one cause is true.



Let's try to understand this technique with some examples:

Situation:

The character in column 1 should be either A or B and in the column 2 should be a digit. If both columns contain appropriate values then update is made. If the input of column 1 is incorrect, i.e. neither A nor B, then message X will be displayed. If the input in column 2 is incorrect, i.e. input is not a digit, then message Y will be displayed.

- A file must be updated, if the character in the first column is either "A" or "B" and in the second column it should be a digit.
- If the value in the first column is incorrect (the character is neither A nor B) then massage X will be displayed.

- If the value in the second column is incorrect (the character is not a digit) then massage Y will be displayed.

Column 1	Column 2
Correct value - A or B	Correct value- Any digit
Incorrect value - Any character except A or B	Incorrect value- Any character except digit

Now, we are going to make a Cause-Effect graph for the above situation:

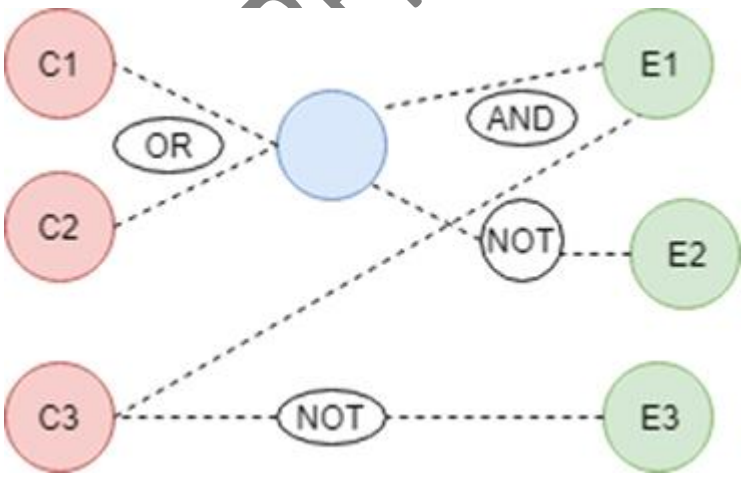
Causes are:

- C1 - Character in column 1 is A
- C2 - Character in column 1 is B
- C3 - Character in column 2 is digit!

Effects:

- E1 - Update made (C1 OR C2) AND C3
- E2 - Displays Massage X (NOT C1 AND NOT C2)
- E3 - Displays Massage Y (NOT C3)

Where AND, OR, NOT are the logical gates.



Effect E1- Update made- The logic for the existence of effect E1 is "**(C1 OR C2) AND C3**". For **C1 OR C2**, any one from C1 and C2 should be true. For logic **AND C3** (Character in column 2 should be a digit), C3 must be true. In other words, for the existence of effect E1 (Update made) any one from C1 and C2

but the C3 must be true. We can see in graph cause C1 and C2 are connected through OR logic and effect E1 is connected with AND logic.

Effect E2 - Displays Massage X - The logic for the existence of effect E2 is "**NOT C1 AND NOT C2**" that means both C1 (Character in column 1 should be A) and C2 (Character in column 1 should be B) should be false. In other words, for the existence of effect E2 the character in column 1 should not be either A or B. We can see in the graph, **C1 OR C2** is connected through NOT logic with effect E2.

Effect E3 - Displays Massage Y- The logic for the existence of effect E3 is "**NOT C3**" that means cause C3 (Character in column 2 is a digit) should be false. In other words, for the existence of effect E3, the character in column 2 should not be a digit. We can see in the graph, **C3** is connected through NOT logic with effect E3.

So, it is the cause-effect graph for the given situation. A tester needs to convert causes and effects into logical statements and then design cause-effect graph. If function gives output (effect) according to the input (cause) so, it is considered as defect free, and if not doing so, then it is sent to the development team for the correction.

E) **Decision table**

Decision table technique is one of the widely used case design techniques for black box testing. This is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form.

That's why it is also known as a cause-effect table. This technique is used to pick the test cases in a systematic manner; it saves the testing time and gives good coverage to the testing area of the software application.

Decision table technique is appropriate for the functions that have a logical relationship between two and more than two inputs.

This technique is related to the correct combination of inputs and determines the result of various combinations of input. To design the test cases by decision table technique, we need to consider conditions as input and actions as output.

Let's understand it by an example:

Most of us use an email account, and when you want to use an email account, for this you need to enter the email and its associated password.

If both email and password are correctly matched, the user will be directed to the email account's homepage; otherwise, it will come back to the login page with an error message specified with "Incorrect Email" or "Incorrect Password."

Now, let's see how a decision table is created for the login function in which we can log in by using email and password. Both the email and the password are the conditions, and the expected result is action.

Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

In the table, there are four conditions or test cases to test the login function. In the first condition if both email and password are correct, then the user should be directed to account's Homepage.

In the second condition if the email is correct, but the password is incorrect then the function should display Incorrect Password. In the third condition if the email is incorrect, but the password is correct, then it should display Incorrect Email.

Now, in fourth and last condition both email and password are incorrect then the function should display Incorrect Email.

In this example, all possible conditions or test cases have been included, and in the same way, the testing team also includes all possible test cases so that upcoming bugs can be cured at testing level.

In order to find the number of all possible conditions, tester uses 2^n formula where n denotes the number of inputs; in the example there is the number of inputs is 2 (one is true and second is false).

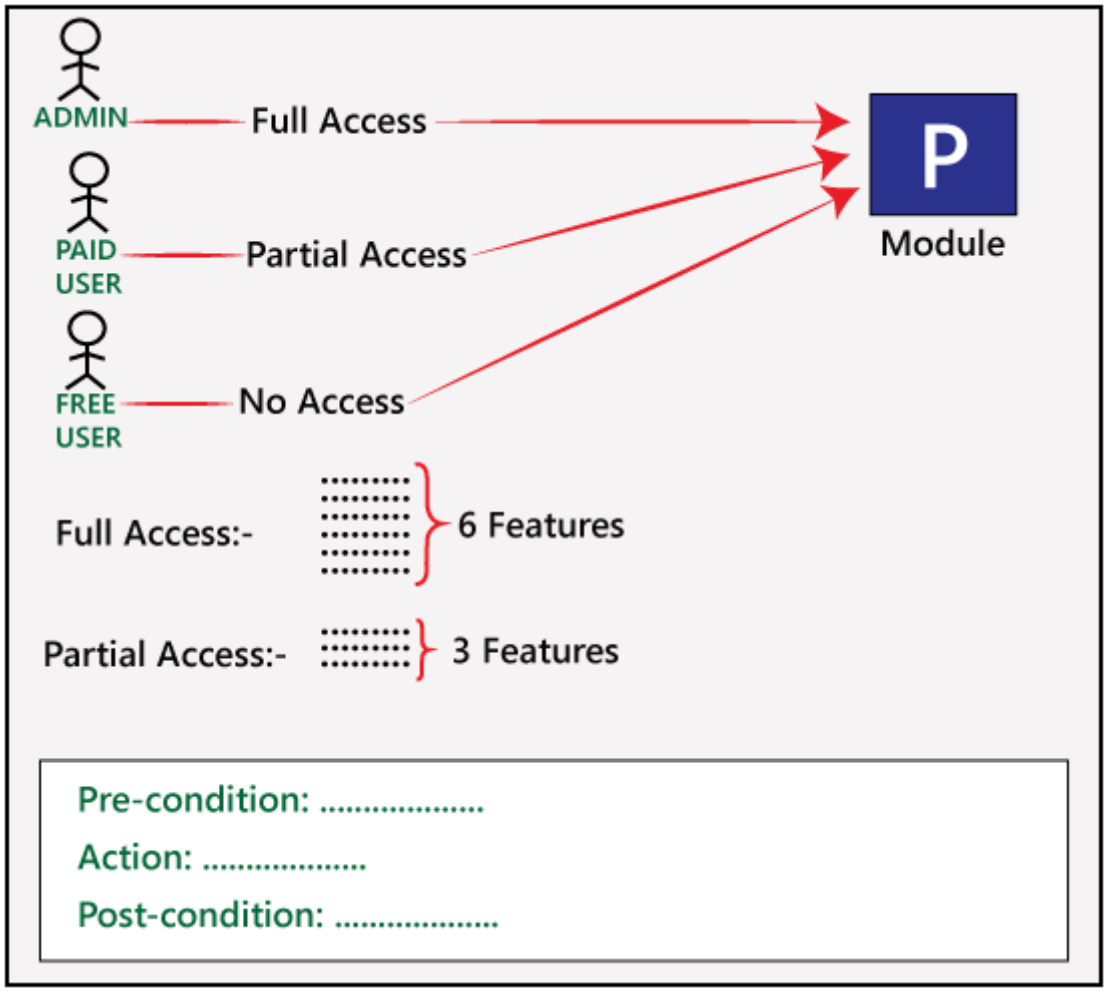
Number of possible conditions = 2^ Number of Values of the second condition
Number of possible conditions =2^2 = 4

While using the decision table technique, a tester determines the expected output, if the function produces expected output, then it is passed in testing, and if not then it is failed. Failed software is sent back to the development team to fix the defect.

F) Use Case Technique

The use case is functional testing of the black box testing used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that can exercise the entire software based on the functionality of each function from start to end.

It is a graphic demonstration of business needs, which describe how the end-user will cooperate with the software or the application. The use cases provide us all the possible techniques of how the end-user uses the application as we can see in the below image, that how the **use case** will look like:



In the above image, we can see that a sample of a use case where we have a requirement related to the customer requirement specification (CRS).

For **module P** of the software, we have six different features.

And here, **Admin** has access to all the **six features**, the **Paid user** has access to the **three features** and for the **Free user**, there is **no access** provided to any of the features.

Like for **Admin**, the different conditions would be as below:

Pre-condition→ Admin must be generated

Action→ Login as Paid user

Post-condition→ 3 features must be present

And for **Free user**, the different condition would be as below:

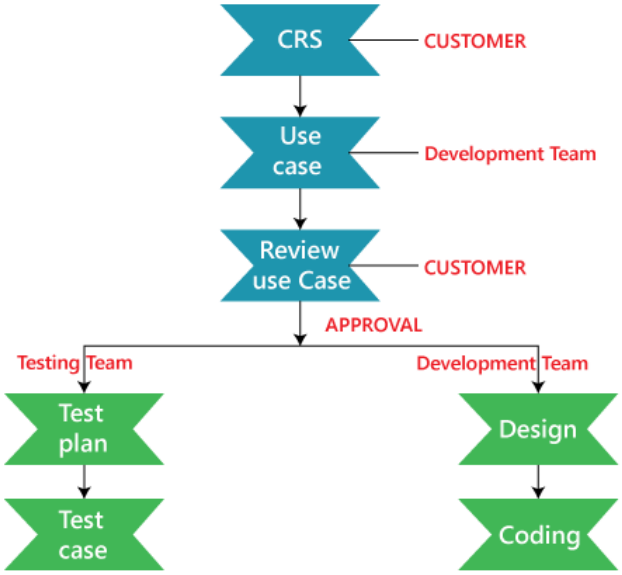
Pre-condition→ free user must be generated

Action→ Login as a free user

Post-condition→ no features

Who writes the use case?

The client provides the customer requirement specification for the application, then the development team will write the **use case** according to the CRS, and the use case is sent to the customer for their review.



If the client approves it, then the approved **use case** is sent to the development team for further design and coding process and these approved use case is also sent to the testing team, so they can start writing the test plan and later on start writing the test cases for the different features of the software.

In the below scenario, there is a tester who represents the user to use the functions of a system one by one. In this scenario, there is an actor who represents the user to use the functions of a software system.

This describes step-by-step functionality of the software application which can be understood with an example, assume that there is a software application of online money transfer. The various steps for transferring money are as follows:

- The user does login for the authentication of the actual user.
- The system checks ID and password with the database to ensure that whether it is a valid user or not.
- If the verification succeeds, the server connects the user to the account page, otherwise returns to the login page.
- In the account page, there are several options because the examiner is checking the money transfer option; the user goes into the money transfer option.

- After successful completion of this step, the user enters the account number in which he wants to transfer money. The user also need to enter other details like bank name, amount, IFSC code, home branch, etc.

In the last step, if there is a security feature that includes verification of the ATM card number and PIN, then enter the ATM card number, PIN and other required details.

If the system is successfully following all the steps, then there is no need to design test cases for this function. By describing the steps to use, it is easy to design test cases for software systems.

How developers develop the use cases

The developers use the standard symbols to write a use case so that everyone will understand easily. They will use the **Unified modeling language** (UML) to create the use cases.

There are various tools available that help to write a use case, such as **Rational Rose.** This tool has a predefined UML symbols, we need to drag and drop them to write a use case, and the developer can also use these symbols to develop the use case.

Advantage of Use Case Technique

The use case technique gives us some features which help us to create an application.

Following are the benefits of using the use case technique while we are developing the product:

- The use case is used to take the functional needs of the system.
- These are the classification of steps, which describe the connections between the user and its actions.
- It starts from an elementary view where the system is created first and primarily used for its users.
- It is used to determine the complete analyses, which help us to achieve the complication, and then it focuses on the one detailed features at a time.