Ex.No:01 Date:	Perform data flow testing for any C program to verify the def.-use variables (Ex: largest of two numbers)

Aim:

To perform data flow testing for any C program to verify the def.-use variables (Ex: largest of two numbers).

Procedure:

Step 1: Start the program

Step 2: Declaring variables as int x,y,a

Step 3: Conditioning the statement as x is greater than y

Step 4: Assigning the statement as a=x+1

Step 5: Else/or, assigning the statement as a=y-1

Step 6: Print a

Step 7: Stop the program

Program:

```
#include<stdio.h>
int main()
{
int x,y,a;
x=20;
y=10;
if(x>y){
a=x+1;
printf("X is greater than Y");
}
else
{
a=y-1;
printf("X is less than Y:%d",&a);
}
return 0;
}
```

Explanation:

Data flow testing is a white-box testing technique that focuses on the flow of data within a program. In this technique, you analyze the paths through which data moves within the program to verify if variables are defined (def.) and used (use) correctly. Let's perform data flow testing for a simple C program that calculates the largest of two numbers.

We'll identify def.-use pairs in this program:

1. **Defining Variables:**

- `int x, y, a;` - These variables are defined at the beginning of the `main` function.

2. **Using Variables:**

- ` printf("X is greater than Y");` - Printing statement that X is greater than Y

- `if (x > y) { a=x+1;}` - `x` is used in the condition, and if true, `a` is assigned the value of `x`.

- `else { a=y-1; }` - `y` is used if the condition is false, and `a` is assigned the value of `y`.

- ` printf("X is less than Y:%d",&a);` - ` X is less than Y ` is used in the `printf` statement.

Based on the def-use pairs, we can create test cases to ensure that variables are defined and used correctly. Here are some test cases:

1. Test Case 1: Positive Path (x is larger):

- Input: `x = 5`, `y = 3`

- Expected Output: "The largest number is 5"

- Explanation: This test case exercises the path where `x` is greater than `y`, ensuring that `max` is assigned correctly.

2. Test Case 2: Negative Path (y is larger):

- Input: `x = 2`, `y = 8`

- Expected Output: "The largest number is 8"

- Explanation: This test case exercises the path where `y` is greater than `x`, ensuring that `max` is assigned correctly.

3. Test Case 3: Equal Numbers:

- Input: `x = 4`, `y = 4`

- Expected Output: "The largest number is 4"

- Explanation: This test case verifies that the program handles the case where `x` and `y` are equal.

4. Test Case 4: Input Validation:

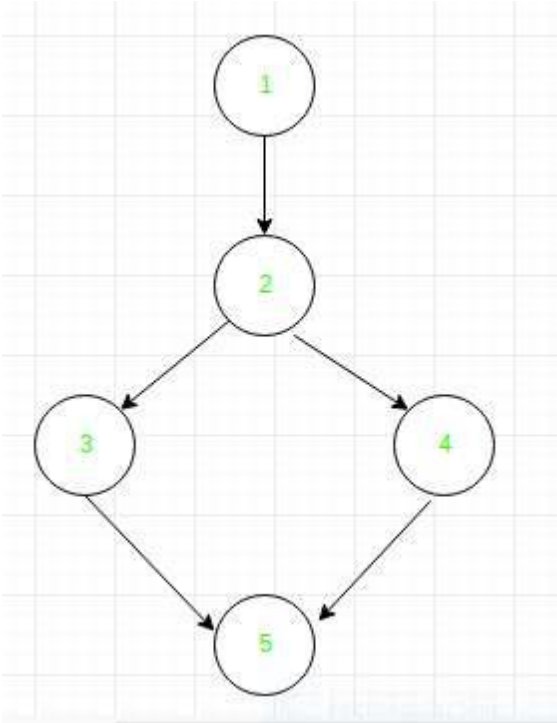
- Input: Invalid input (e.g., non-integer input)

- Expected Output: Error handling for invalid input

- Explanation: This test case checks how the program handles invalid input and ensures that it doesn't cause unexpected issues with variable definitions and usages.

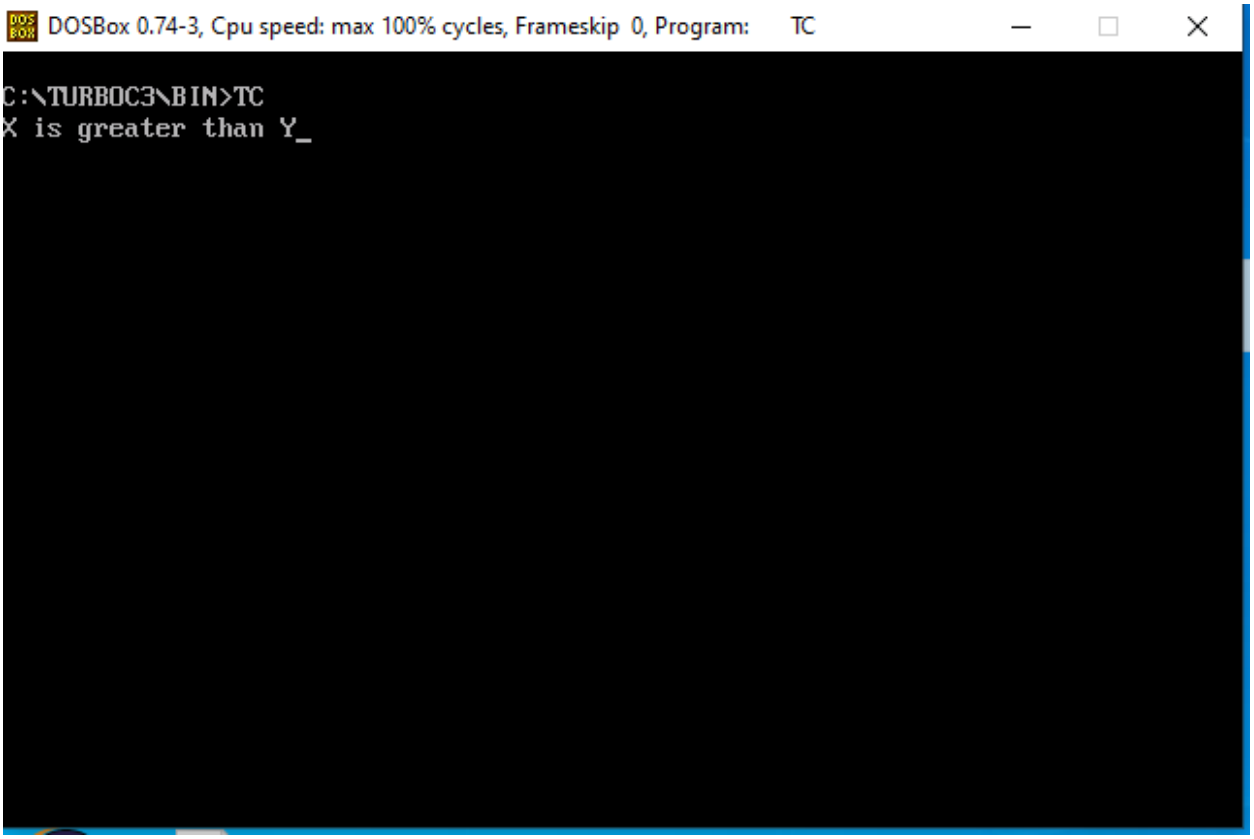
By designing and executing these test cases, you can verify that the program correctly defines and uses variables according to the def.-use relationships in the code.

Output:



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



Result:

Hence the above program was executed successfully and verified.

Ex.No:02 Date:	Using Selenium IDE, Write a test suite containing minimum 4 test cases for any simple C program (Ex: To check Adam Number)

Aim:

To write a test suite containing minimum 4 test cases for any simple C program (Ex: To check Adam number)

Procedure:

Step 1: Click recording options in selenium ide Firefox.

Step 2: Create a project name as pname.

Step 3: Copy the URL in the dialog box and click start recording.

Step 4: Create a test suite in left top corner and click stop recording.

Step 5: Create a test name as tname.

Step 6: Click Run button

Step 7: Write a C program, Find reverse of number.

Step 8: Find square of number in Step 2.

Step 9: Find reverse of number in Step 3.

Step 10: Find whether number obtained in Step 4 and Step 1 are equal or not.

Definitions:

1. **Test Cases**

A Test Case is a set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly. The process of developing test cases can also help find problems in the requirements or design of an application.

2. **Test suite**

In software development, a test suite, less commonly known as a 'validation suite', is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviors.

3. **Selenium IDE**

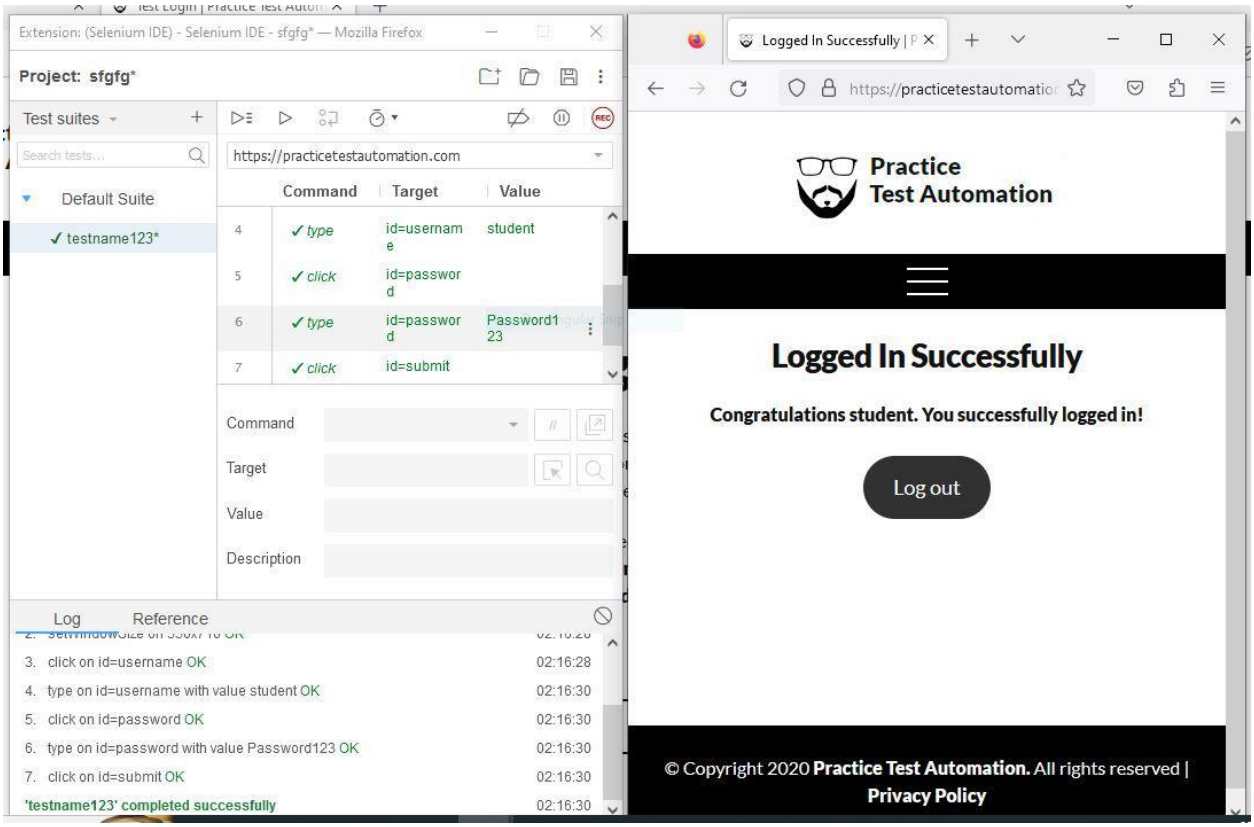
The Selenium-IDE (Integrated Development Environment) is the tool you use to develop your Selenium test cases. It’s an easy-to-use Chrome and Firefox extension and is generally the most efficient way to develop test cases. It records the user’s actions in the browser for you, using existing Selenium commands, with parameters defined by the context of that element. This is not only a time-saver, but also an excellent way of learning Selenium script syntax.

Program:

```
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class DefaultSuiteTest {
private WebDriver driver;
```

```
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
public void setUp() {
driver = new FirefoxDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
public void tearDown() {
driver.quit();
}
@Test
public void testname123() {
driver.get("https://practicetestautomation.com/practice-test-login/");
driver.manage().window().setSize(new Dimension(550, 710));
driver.findElement(By.id("username")).click();
driver.findElement(By.id("username")).sendKeys("student");
driver.findElement(By.id("password")).click();
driver.findElement(By.id("password")).sendKeys("Password123");
driver.findElement(By.id("submit")).click();
}
}
```

Output:



Result:

Hence the above program was executed successfully and verified.

Ex.No:03 Date:	Write and test a program to update 10 student records into tables into Excel file. (Selenium)

Aim:

To Write and test a program to update 10 student records into tables into Excel file. (Selenium)

Procedure:

Step 1: Start a program

Step 2: Import files to upload

Step 3: Import panda library as pd

Step 4: Import io library

Step 5: Assigning the statement as

‘df = pd.read_csv(io.BytesIO(uploaded['1.csv']))’

Step 6: Print the statement with the variable ‘df’.

Step 7: Import panda library as pd

Step 8: Reading the csv files ‘df = pd.read_csv("1.csv")’

Step 9: Updating the column value/data ‘df.loc[5, 'Name'] = 'SHIV CHANDRA'’

Step 10: Writing into the file ‘df.to_csv("1.csv", index=False)’

Step 11: Print the statement with the variable ‘df’.

Program:

```
from google.colab import files

uploaded = files.upload()

import pandas as pd

import io

df = pd.read_csv(io.BytesIO(uploaded['1.csv']))

print(df)

import pandas as pd

# reading the csv file

df = pd.read_csv("1.csv")

# updating the column value/data

df.loc[5, 'Name'] = 'SHIV CHANDRA'

# writing into the file

df.to_csv("1.csv", index=False)

print(df)
```

Output:

Read

	Rollno	Name	DBMS	JAVA	LINUX	Total
0	1	Anurang	30	65	30	125
1	2	Amit	45	46	56	147
2	3	Rahul	50	30	30	110
3	4	John	68	25	56	149
4	5	Sunder	99	68	49	216
5	6	Yasmeen	79	90	65	234
6	7	Sherina	89	89	36	214
7	8	Nitish	53	67	45	165
8	9	Tanzil	41	62	72	175
9	10	Jayant	50	75	31	156

Updated

	Rollno	Name	DBMS	JAVA	LINUX	Total
0	1	Anurang	30	65	30	125
1	2	Amit	45	46	56	147
2	3	Rahul	50	30	30	110
3	4	John	68	25	56	149
4	5	Sunder	99	68	49	216
5	6	SHIV CHANDRA	79	90	65	234
6	7	Sherina	89	89	36	214
7	8	Nitish	53	67	45	165
8	9	Tanzil	41	62	72	175
9	10	Jayant	50	75	31	156

Result:

Hence the above program was executed successfully and verified.

Ex.No:04 Date:	**Write and test a program to select the number of students who have scored more than 60 in any one subject (Or all subjects). (Selenium)**

Aim:

To Write and test a program to select the number of students who have scored more than 60 in any one subject (or all subjects). (Selenium)

Procedure:

Step 1: Start the program.

Step 2: Assigning the values using ‘students’ variable.

Step3: Define ‘count_student_above_60’ function with the parameter of data and subject.

Step4: Assign count =0

Step5: In the for loop ‘students’ variable was assigned with data parameter.

Step6: In If conditional statement ‘students’ variable was assigned with parameter subject and it was assigned ‘greater than’ 60.

Step7: Assigned incremental operator as count+=1.

Step8: Return count.

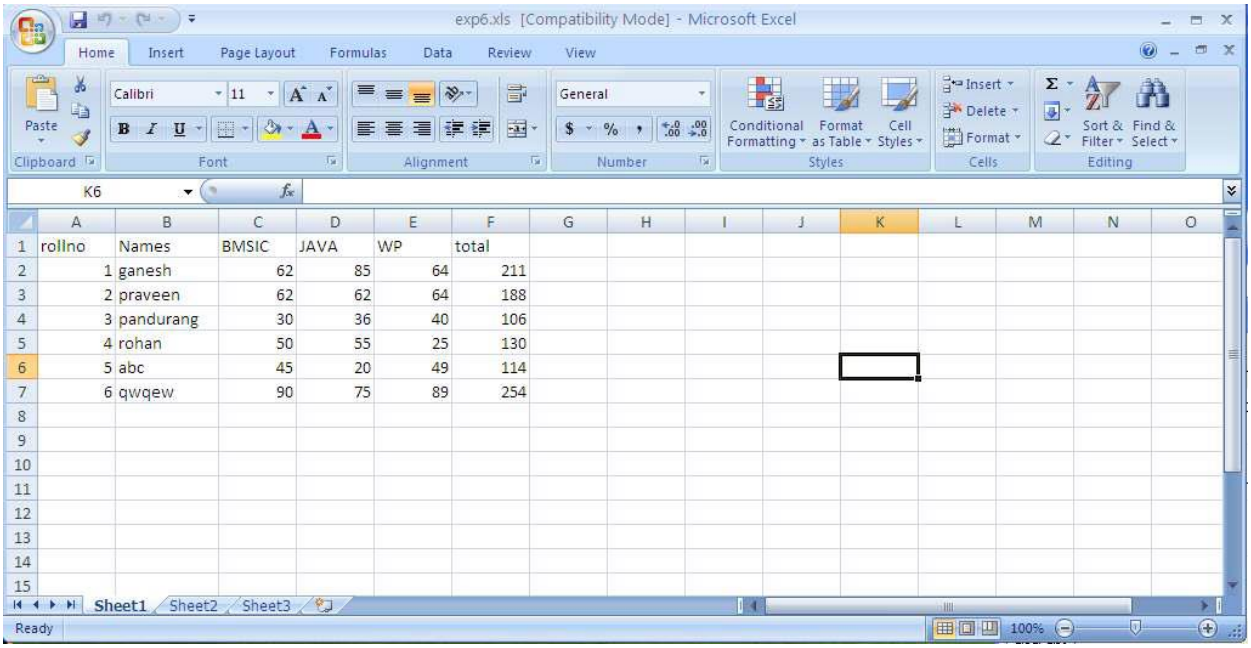
Step9: Stop the program.

Program:

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import jxl.Sheet;
import jxl.Workbook;
import jxl.write.Label;
import jxl.write.WritableSheet;
import jxl.write.WritableWorkbook;
import org.testng.annotations.*;
public class exp7 {
@BeforeClass
public void setUp() throws Exception {
}
@Test
public void testImportexport1() throws Exception {
FileInputStream fi = new FileInputStream("D:\\exp6.xls");
Workbook w = Workbook.getWorkbook(fi);
Sheet s = w.getSheet(0);
String a[][] = new String[s.getRows()][s.getColumns()];
FileOutputStream fo = new FileOutputStream("D://exp7Result.xls");
WritableWorkbook wwb = Workbook.createWorkbook(fo);
WritableSheet ws = wwb.createSheet("result", 0);
int c=0;
for (int i = 0; i < s.getRows(); i++) {
for (int j = 0; j < s.getColumns(); j++)
```

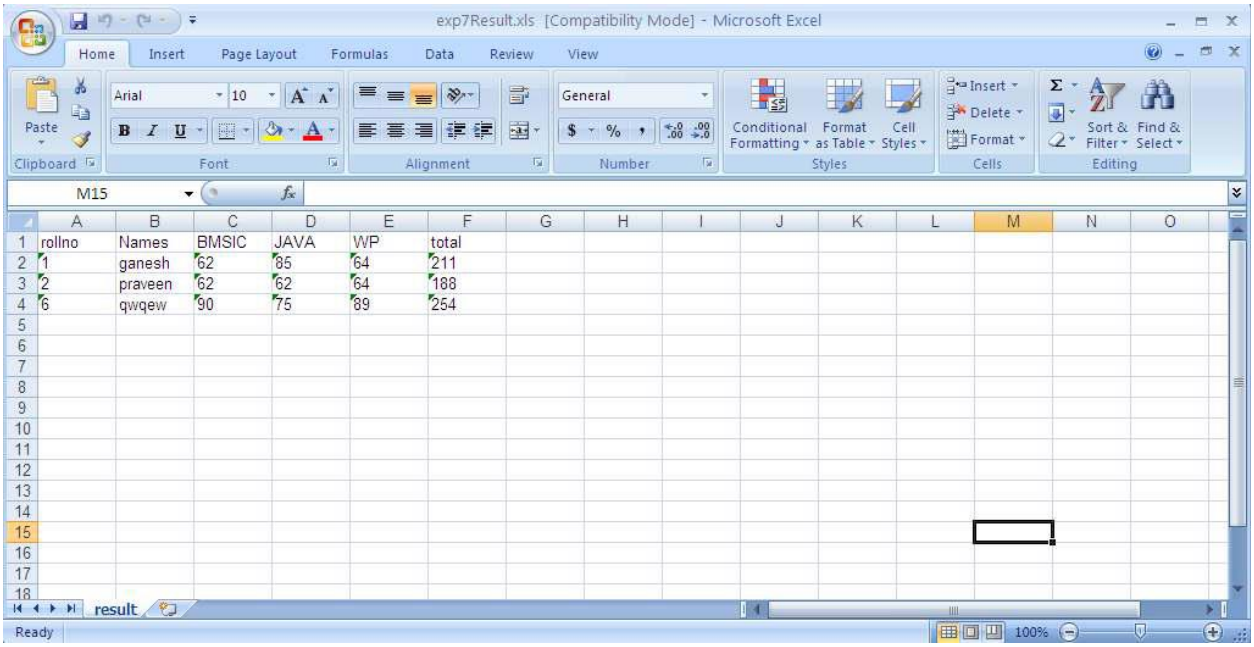
```
{
if(i >= 1)
{
String b= new String();
b=s.getCell(3,i).getContents();
int x= Integer.parseInt(b);
if( x < 60)
{
c++;
break;
}
}
a[i][j] = s.getCell(j, i).getContents();
Label l2 = new Label(j, i-c, a[i][j]);
ws.addCell(l2);
}
}
wwb.write();
wwb.close();
}
}
```

Input:



rollno	Names	BMSIC	JAVA	WP	total
1	ganesh	62	85	64	211
2	praveen	62	62	64	188
3	pandurang	30	36	40	106
4	rohan	50	55	25	130
5	abc	45	20	49	114
6	qwqew	90	75	89	254

Output:



rollno	Names	BMSIC	JAVA	WP	total
1	ganesh	62	85	64	211
2	praveen	62	62	64	188
6	qwqew	90	75	89	254

Result:

Hence the above program was executed successfully and verified.

Ex.No:05 Date:	Write and test a program to login to a specific web page

Aim:

To Write and test a program to login to a specific web page

Procedure:

Step1: Click recording options in selenium ide Firefox.

Step2: Create a project name as pname1.

Step3: Copy the URL in the dialog box and click start recording.

Step4: Click stop recording.

Step5: Create a test name as tname1.

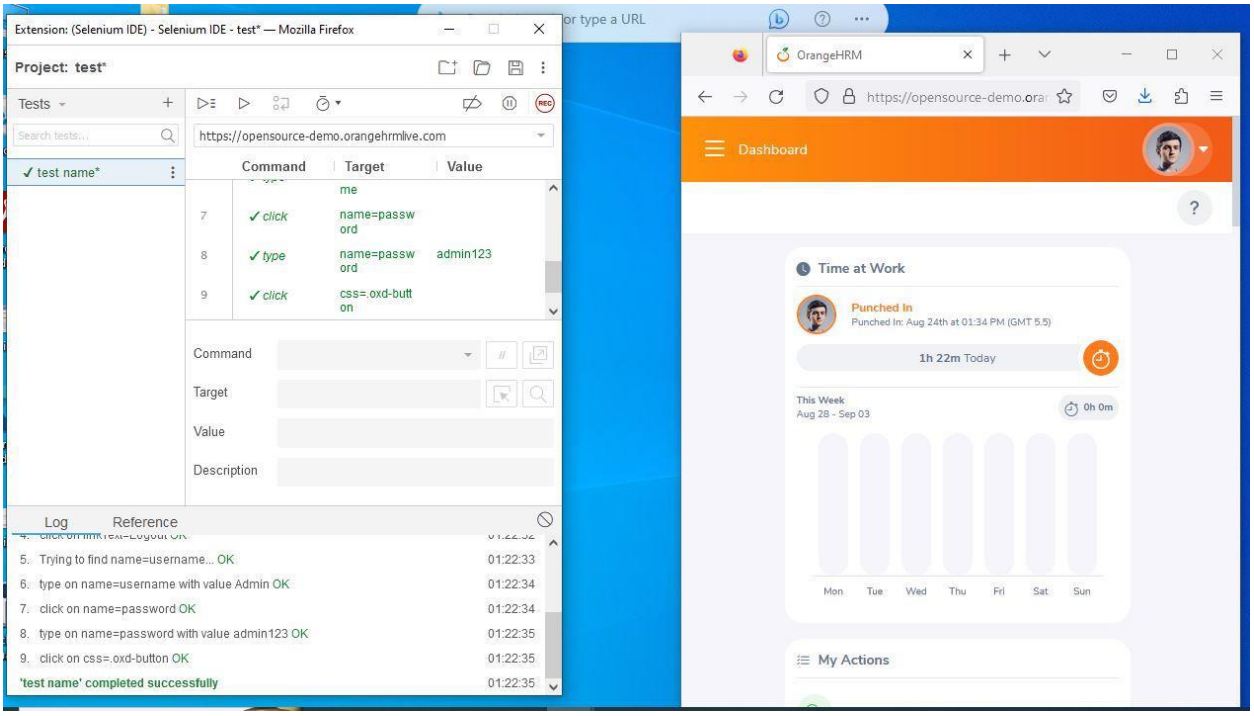
Step6: Click Run button

Program:

```
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class TestnameTest {
private WebDriver driver;
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
```

```
public void setUp() {
driver = new FirefoxDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
public void tearDown() {
driver.quit();
}
@Test
public void testname() {
driver.get("https://opensource-demo.orangehrmlive.com/web/index.php/dashboard/index");
driver.manage().window().setSize(new Dimension(578, 697));
driver.findElement(By.cssSelector(".bi-caret-down-fill")).click();
driver.findElement(By.linkText("Logout")).click();
driver.findElement(By.name("username")).click();
driver.findElement(By.name("username")).sendKeys("Admin");
driver.findElement(By.name("password")).click();
driver.findElement(By.name("password")).sendKeys("admin123");
driver.findElement(By.cssSelector(".oxd-button")).click();
}
}
```

Output:



Result:

Hence the above program was executed successfully and verified.

Ex.No:06 Date:	Write and test a program to provide a total number of objects present / available on the page. (Selenium)

Aim:

To Write and test a program to provide a total number of objects present / available on the page. (Selenium)

Procedure:

Step 1: Click recording options in selenium ide Firefox.

Step2: Paste the URL and click start recording.

Step3: Click empty row and choose ‘verify element present’ and type ‘linkText=About’

Step4: Click find button.

Step5: Stop recording

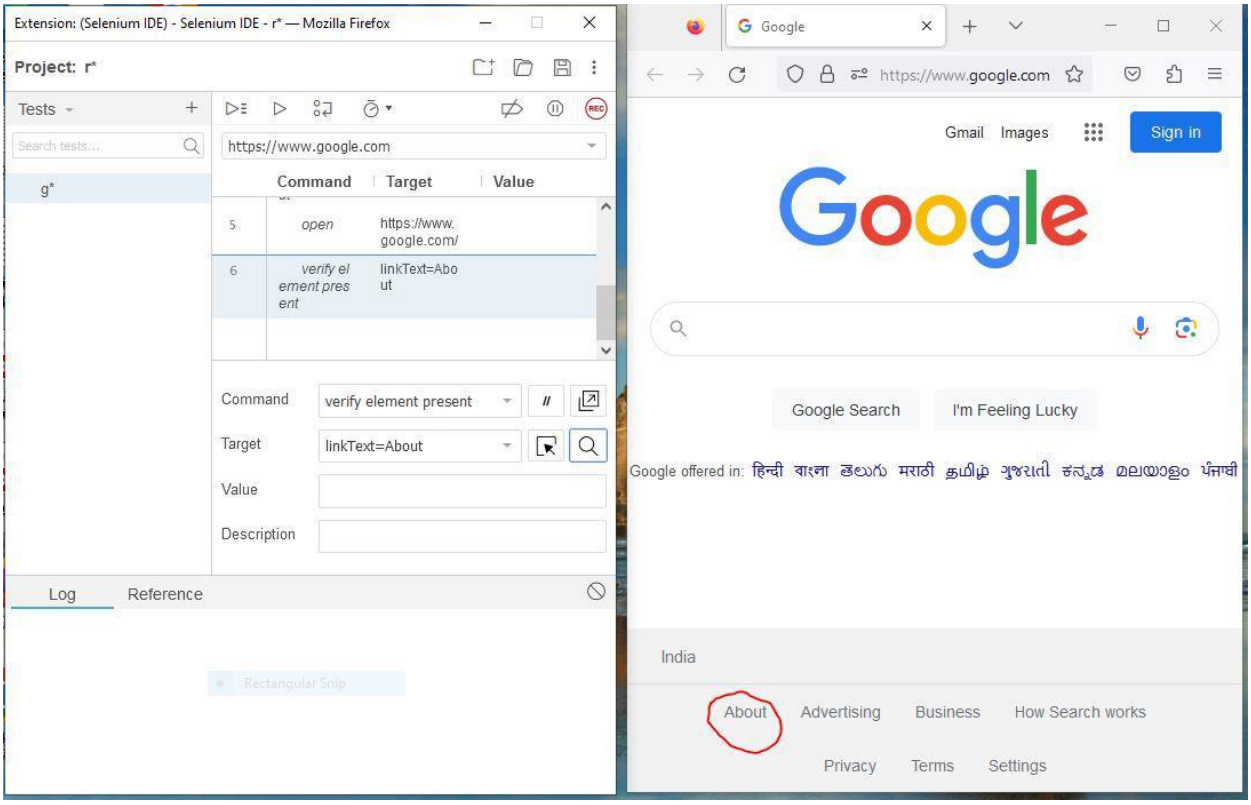
Step5: Save the test and run the test.

Program:

```
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class GTest {
private WebDriver driver;
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
```

```
public void setUp() {
driver = new FirefoxDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
public void tearDown() {
driver.quit();
}
@Test
public void g() {
driver.get("https://www.google.com/");
driver.manage().window().setSize(new Dimension(550, 698));
{
WebElement element = driver.findElement(By.cssSelector(".gb_d"));
Actions builder = new Actions(driver);
builder.moveToElement(element).perform();
}
{
WebElement element = driver.findElement(By.tagName("body"));
Actions builder = new Actions(driver);
builder.moveToElement(element, 0, 0).perform();
}
driver.get("https://www.google.com/");
{
List<WebElement> elements = driver.findElements(By.linkText("About"));
assert(elements.size() > 0);
}
}
}
```

Output:



Total Number of links = 41

Total Number of buttons = 1

Total Number of input fields = 3

Result:

Hence the above program was executed successfully and verified.

Ex.No:07 Date:	Write and test a program to get the number of list items in a list / combo box. (Selenium)

Aim:

To Write and test a program to get the number of list items in a list / combo box. (Selenium)

Procedure:

Step1: Click and open selenium IDE.

Step2: Click ‘create new project’.

Step3: Give a project name.

Step4: Click on the record button and paste URL.

Step5: Click somewhere and action will be recorder.

Step6: Save test case name.

Step7: Click in the empty row and in a command type ‘select’.

Step8: Type target ‘id=demobasic’ and value=One.

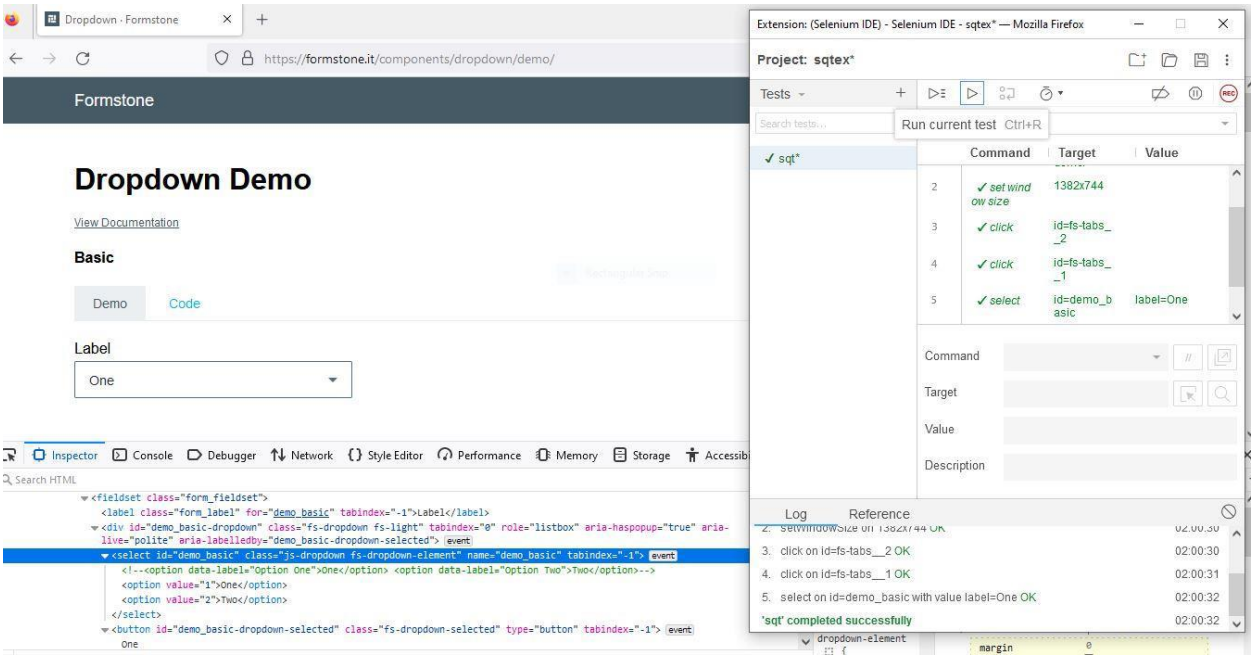
Step9: Finally click run button.

Program:

```
// Generated by Selenium IDE
import org.junit.Test;
import org.junit.Before;
import org.junit.After;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.is;
import static org.hamcrest.core.IsNot.not;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.Dimension;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.Alert;
import org.openqa.selenium.Keys;
import java.util.*;
import java.net.MalformedURLException;
import java.net.URL;
public class SqtTest {
private WebDriver driver;
private Map<String, Object> vars;
JavascriptExecutor js;
@Before
```

```
public void setUp() {
driver = new FirefoxDriver();
js = (JavascriptExecutor) driver;
vars = new HashMap<String, Object>();
}
@After
public void tearDown() {
driver.quit();
}
@Test
public void sqt() {
driver.get("https://formstone.it/components/dropdown/demo/");
driver.manage().window().setSize(new Dimension(1382, 744));
driver.findElement(By.id("fs-tabs__2")).click();
driver.findElement(By.id("fs-tabs__1")).click();
{
WebElement dropdown = driver.findElement(By.id("demo_basic"));
dropdown.findElement(By.xpath("//option[. = 'One']")).click();
}
}
}
```

Output:



Result:

Hence the above program was executed successfully and verified.

Ex.No:08 Date:	**Identify system specification and design test cases to test any application using any one of a testing tool (Selenium/Bugzilla/Test Director)**

Aim:

To design test cases to test any application using any one of a testing tool (Selenium/Bugzilla/TestDirector).

Procedure:

Step1: Install selenium server

Step2: Command prompt type “java -version”

Step3: Click project->new project.

Step 4: Right click src and choose package option and name it as “sample app”

Step 5: Right “sample app” choose class option and name it as “HelloWorld”

Step 6: Start writing a program

Step 7: Download selenium grid server “selenium-server-4.12.0.jar”

Step8: In intellij, file->project structure->modules

Step9: Press ‘+’ symbol and choose jar or directories and fetch selenium server

Step10: Click on apply and ok.

Step11: Write a code

Step12: Download chromedriver

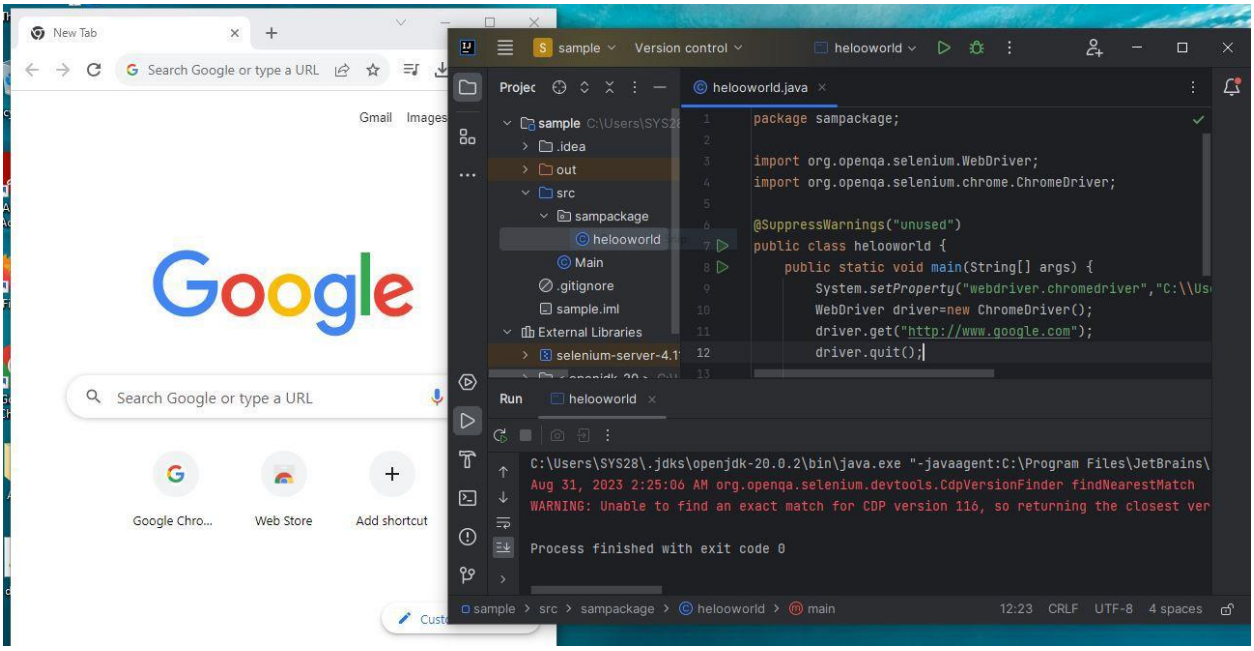
Step13: Right click chromedriver->copypath and paste in a code

Step14: Run a code.

Program:

```
package sampackage;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
@SuppressWarnings("unused")
public class helooworld {
public static void main(String[] args) {
System.setProperty("webdriver.chromedriver","C:\\Users\\SYS28\\Downloads\\chr
omedriver_win32");
WebDriver driver=new ChromeDriver();
driver.get("http://www.google.com");
driver.quit();
}
}
```

Output:



Result:

Hence the above program was executed successfully and verified.

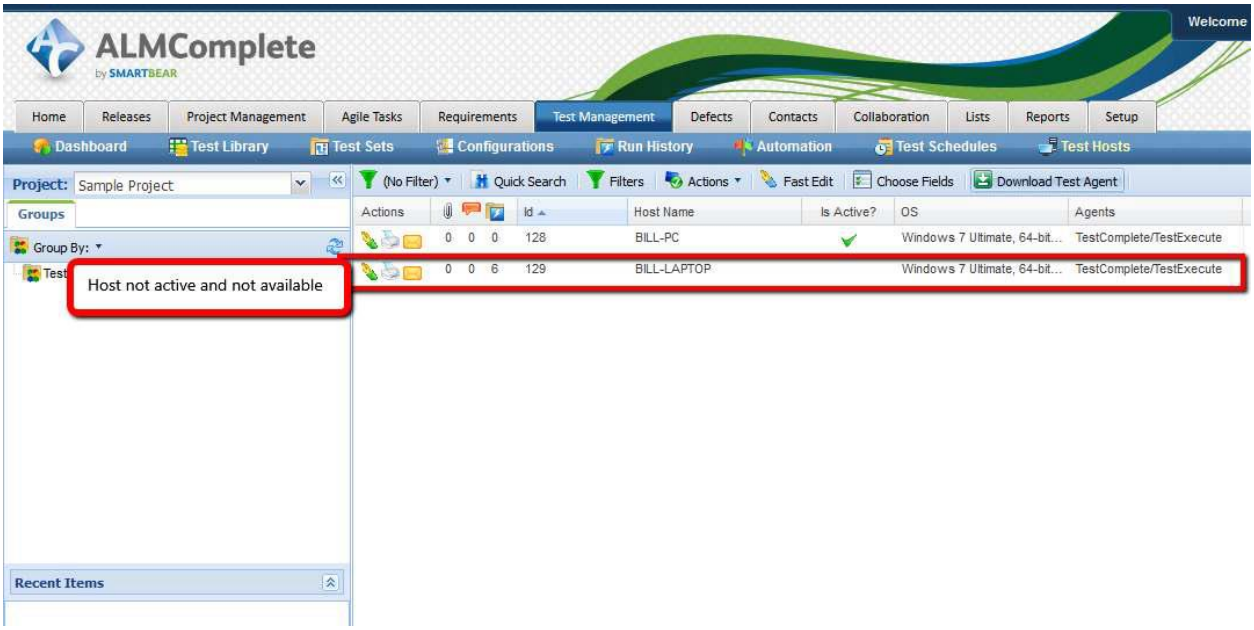
Ex.No:09 Date:	Automate the test cases of the system using any test automation tool (Bugzilla /QA Complete)

Aim:

To create the Automate test cases using test automation tool (using QA Complete)

Procedure:

Setting up automated tests in QA complete



Step 1: Check in QA Complete that the Test Complete host is available by viewing the 'Test Hosts' records. If the host isn't listed at all then enable the 'Show Inactive Test Hosts' option. If the host isn't active then start the service on the Test Complete machine.

Step 2: On the Test Complete machine Press Ctrl+Shift+Esc to display task manager and then click on the 'Services' tab followed by the 'Services' button. In the Services window click on the 'Test Manager Agent' service and start the service.

Step 3: Creating an Automate Test

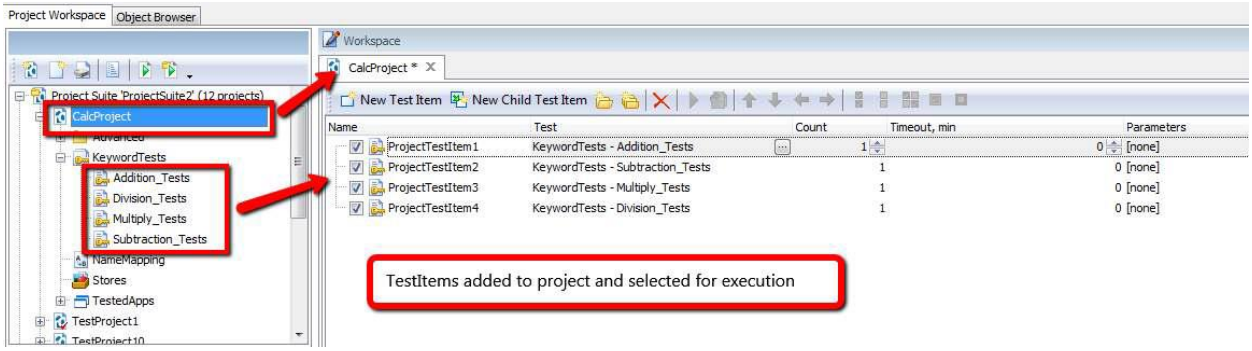
This is a 4 stage process.

1. Package up the Test Complete project suite
2. Define the Automated Test in the QA Complete Test Library
3. Execution of Automated Tests – standalone
4. Execution of Automated Tests – as part of a Test Set

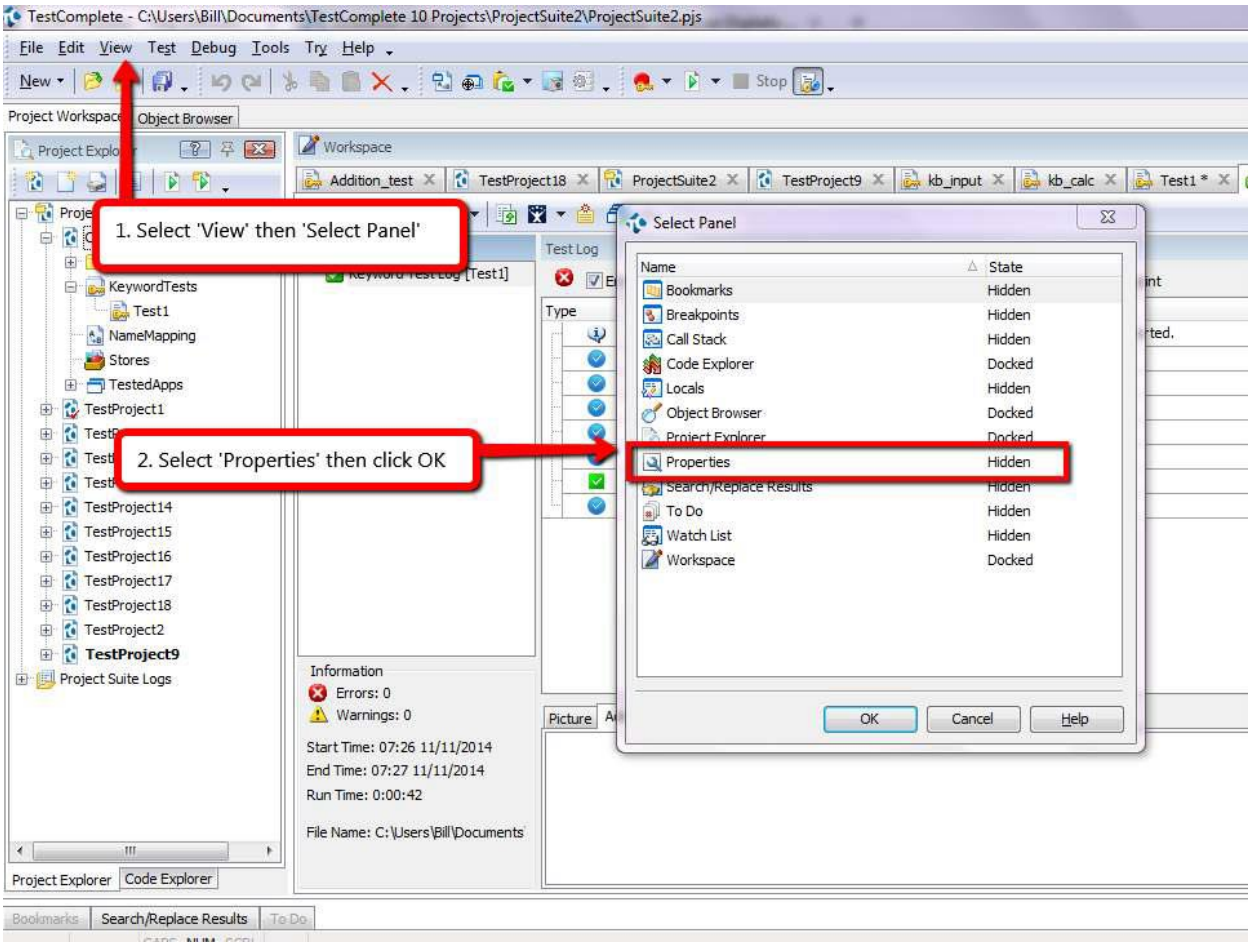
1: Package up the Test Complete Project Suite

To zip the project suite up follow these steps:

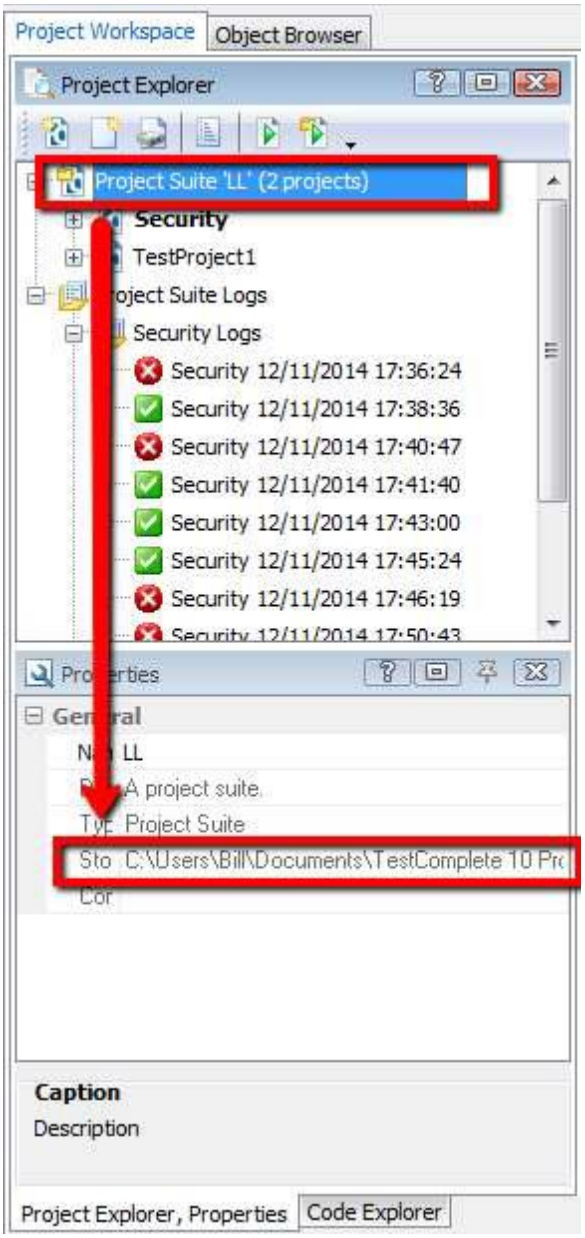
1) Make sure to define the ‘Test Items’ and enabled them within Test Complete project(s)



2) Find the location of test complete project suite on the file system of test complete machine



From here we can see where test complete is storing the project on file system.



3) On the file system (or in test complete) remove the log files.

4) At the project suite level on the file system find the folder containing project suite and zip up this project suite.

Define the Automated Test in the QA Complete Test Library:

Create the test case in the 'Test Library' area of QA Complete and then attach the zipped up

Test Complete project suite to this test case.

5) First we need to create a new test. Navigate to the Test Management Library area in QA Complete and select 'Add New. Then we need to define the usual Meta data required to create the test case (e.g. Title, Description, etc.). A couple of fields that are important though:

Execution Type: set this to Automated

Default Host Name: set this to the host that will be used by default to execute the automated test

Assuming selected Execution Type = Automated then save the test case to the 'Automations' tab for the test case. Click 'Add New' to add a new Test Complete Project Suite.

When adding a new Test Complete project suite to QA Complete following 6 fields will be presented:

Title: either leaves this blank and QA Complete will give this automated test the same name as the Test Complete project or define your own name

Time Out: this is how long it should take to run the test. If it goes past this time out value then the test runner will stop running the test and move on to the next one.

Entry Point: use this to identify a specific test or project to run. If this field is blank the whole project suite will be run. Specify a specific test case to execute an individual test case or a specific project to run only one project.

Agent: at the moment QA Complete only supports one type of test agent which is

Test Complete/Test Execute. Other types of test agent are in the pipeline.

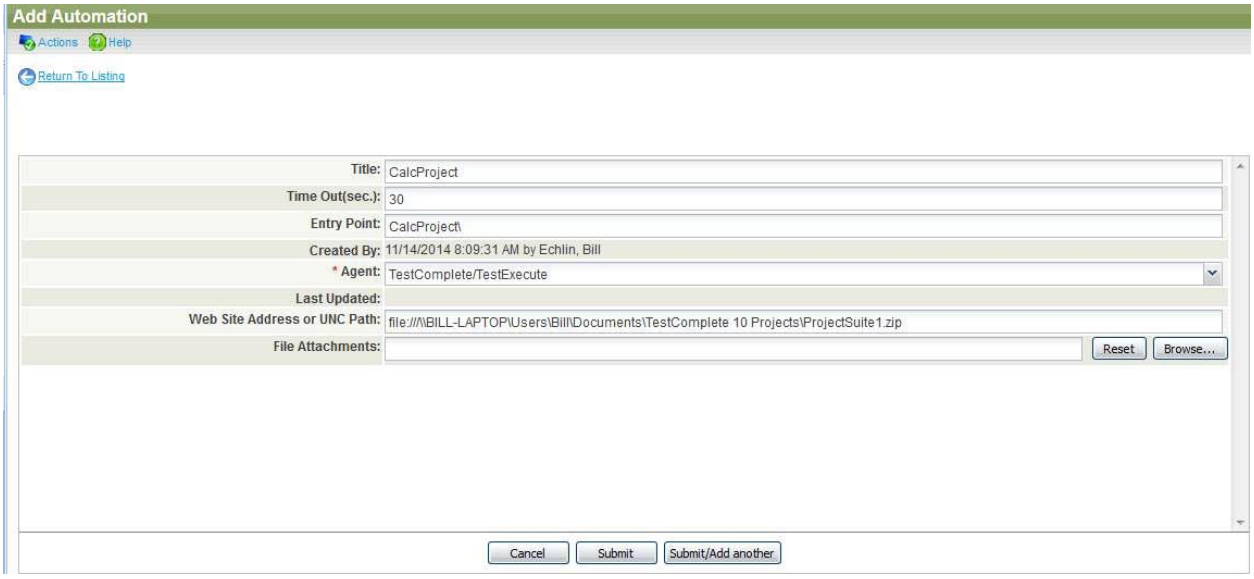
Web Site Address or UNC Path: place the zipped up project suite file on a shared drive. In which case, define the path to that location and the file name here. Alternatively…

File Attachments: attached the zipped up project suite file to the QA Complete test case and upload the file to QA Complete

A completed record with a project suite zip file uploaded looks like this (the entry point in this example is at the Test Case level

A completed record with a UNC path looks like this (The Entry point in this example is at the project level)…

Output:



At this point in time, only add one automation project suite to a single QA Complete automated test case

Result:

Hence the above program was executed successfully and verified.

Ex.No:10 Date:	Design test cases for web pages to test any web sites (Web Performance Analyzer/Open STA)

Aim:

To design test cases for web pages to test any websites.

Procedures:

Step1: Type webpage test in the search engine.

Step2: Paste particular URL in the webpage test website.

Step3: Click test button.

Step4: Finally the web performance analysis report is generated.

Output: Webpage test (Web performance analyzer)

URL: https://www.tamildailycalendar.com/tamil_daily_calendar.php
From: Virginia USA - EC2 - **Chrome - Emulated Motorola G (gen 4) - 4G**

- Test runs: 3
- Connectivity: 9000/9000 Kbps, 170ms Latency
- Custom Metrics

Page Performance Metrics(Run 1)

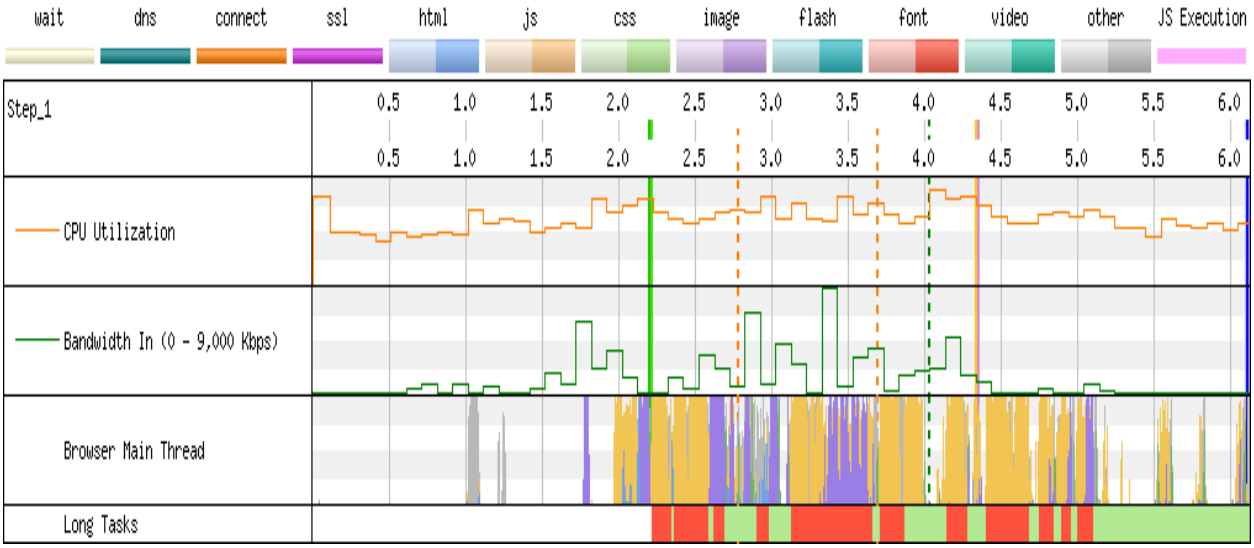
First View (Run 1)

TTFB	Start Render	DC Time	DC Requests	DC Bytes	Total Time	Total Requests	Page Weight
.000s	.000s	.000s	-	-	.000s	-	-

Waterfall View

Start Render Document Complete

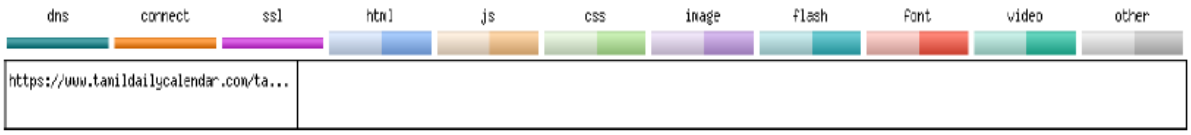
Render Blocking Resource Insecure Request 3xx response 4xx+ response Doesn't Belong to Main Doc



customize waterfall • View all Images • View HTTP/2 Dependency Graph • Filmstrip

Connection View

DNS Lookup Initial Connection Start Render Document Complete



Request Details

Before Start Render Before On Load After On Load 3xx Response 4xx Response

Request Details

#	Resource	Content Type	Request Start	DNS Lookup	Initial Connection	Time to First Byte	Content Download	Bytes Downloaded	CPU Time	Error/Status Code	IP

Result:

Hence the above program was executed successfully and verified.