

**Software Testing**

**EPAM**

**INDIVIDUAL PRACTICAL TASK REPORT**

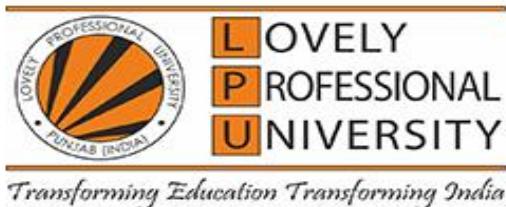
Module Framework Practical Task Report

**Bachelor Of Technology**

**Computer Science & Engineering**

**(Software Testing)**

**LOVELY PROFESSIONAL UNIVERSITY PHAGWARA,  
PUNJAB**



01/05/2023

SUBMITTED BY

Name of the student :- Nitu Singh

Registration Number :- 11909278

Roll No:- 14

SUBMITTED TO

Name of the assistant professor:- Sakshi

## Step 1:- Module 3 code

The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_project\_11909278' open. The code editor displays the `Google_Price_Cal_Test.java` file. The code is a test class for a price calculation application, using Selenium WebDriver and TestNG annotations. The right-hand panel shows the Maven lifecycle and dependencies. The status bar at the bottom indicates the date as 02-05-2023.

```
package org.example;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.Assert;
import org.testng.annotations.AfterClass;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Parameters;
import org.testng.annotations.Test;

public class Google_Price_Cal_Test {
    WebDriver driver;
    Google_Price_Calculate_App obj;
    String VM_Of_Class;
    String region;
    String S_S_D;
    String No_of_Instance;
}
```

## Step 2 :- Module 3 code

The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_project\_11909278' open. The code editor displays the `Google_Price_Calculate_App.java` file. The code defines several utility methods for interacting with a web application, such as finding commitment terms and instance types. The right-hand panel shows the Maven lifecycle and dependencies. The status bar at the bottom indicates the date as 02-05-2023.

```
@FindBy(xpath="//div[normalize-space()='Commitment term: 1 Year']")
WebElement committedTime;

public WebElement Get_Commitment(){
    WebElement time=committedTime;
    return time;
}

@FindBy(xpath="//div[contains (text()),'Instance type: m1-standard-8']")
WebElement instanceData;

public WebElement Get_Instance_Type(){
    WebElement instance=instanceData;
    return instance;
}

@FindBy(xpath="//div[contains (text()),'Local SSD: 2x375 GiB']")
WebElement ssdData;

public WebElement Get_Ssd_Data()
{
    WebElement ssd=ssdData;
    return ssd;
}
```

## Step 3 :- Module 3 Code

The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_project\_11909278' open. The code editor displays the file 'Google\_Price\_Calculate\_App.java'. The code implements several methods to interact with a web application, including clicking on dropdowns and buttons, and performing assertions. The Maven tool window on the right shows various lifecycle stages like clean, validate, compile, test, package, verify, install, site, and deploy.

```
    Data_Center_Location_Drop_Box.click();
    Thread.sleep( millis: 1000 );
    Data_Center_Location.click();
}

1 usage
@FindBy(xpath="//md-select[@placeholder='Committed usage']")
WebElement Committed_Use_Drop_Box;
1 usage
@FindBy(xpath="//md-option[@id='select_option_134']")
WebElement Committed_Use_One_Year;
1 usage
public void Select_Committed_Usage() throws InterruptedException {
    Committed_Use_Drop_Box.click();
    Thread.sleep( millis: 1000 );
    Committed_Use_One_Year.click();
}

1 usage
@FindBy(xpath="//form[@name='ComputeEngineForm']//button[@type='button'][normalize-space()='Add to Estimate']")
WebElement Add_To_Estimate_Button;
1 usage
public void Push_Add_To_Estimate() { Add_To_Estimate_Button.click(); }

1 usage
@FindBy(xpath="//div[normalize-space()='Provisioning model: Regular']")
WebElement Vm_Class_Data;
1 usage
public WebElement Get_Vm_Class_Data()
```

## Step 4 :- Module 3 code

The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_project\_11909278' open. The code editor displays the file 'Google\_Price\_Cal\_Test.java'. It contains a test class with a setup method that handles browser selection and initialization. A test method is present to check information in a VM class string. The Maven tool window on the right shows various lifecycle stages.

```
@BeforeClass
@Parameters({"browser", "Url"})
public void openBrowser(String browser, String Url)
{
    if(browser.equalsIgnoreCase("chrome"))
    {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        obj = new Google_Price_Calculate_App(driver);
    }
    else if(browser.equalsIgnoreCase("firefox"))
    {
        WebDriverManager.firefoxdriver().setup();
        driver = new FirefoxDriver();
        obj = new Google_Price_Calculate_App(driver);
    }
    driver.get(Url);
}

no usages: nitu
@Test
public void checkInformationInVmClassString() throws InterruptedException
{

    driver.manage().window().maximize();
    obj.Number_of_Instances_Field(NumberOfInstances: "4");
}
```

## Step 5 :- Module 3 Code

The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_projectt\_11909278' open. The code editor displays the file 'Google\_Price\_Cal\_Test.java'. The code implements a test method 'check\_Data\_is\_correct' that asserts various properties of a 'Vm' object. The Maven tool window on the right shows the 'Lifecycle' section with several goals listed.

```
Thread.sleep( millis: 5000 );
obj.Add_Gpus_CheckBox();
obj.Select_Type_Of_Gpus();
obj.Select_Number_Of_Gpus();
obj.Select_Local_Ssd();
obj.Select_Data_Center_Location();
obj.Select_Committed_Usage();
obj.Push_Add_To_Estimate();

//Data store from Object

VM_OF_Class = obj.Get_Vm_Class_Data().getText();
region=obj.Get_Location().getText();
S_SD = obj.Get_Ssd_Data().getText();
No_of_Instance = obj.Get_Instance_Type().getText();
Time_Taken = obj.Get_Commitment().getText();
Cost_of_USD = obj.Get_Cost().getText();

}

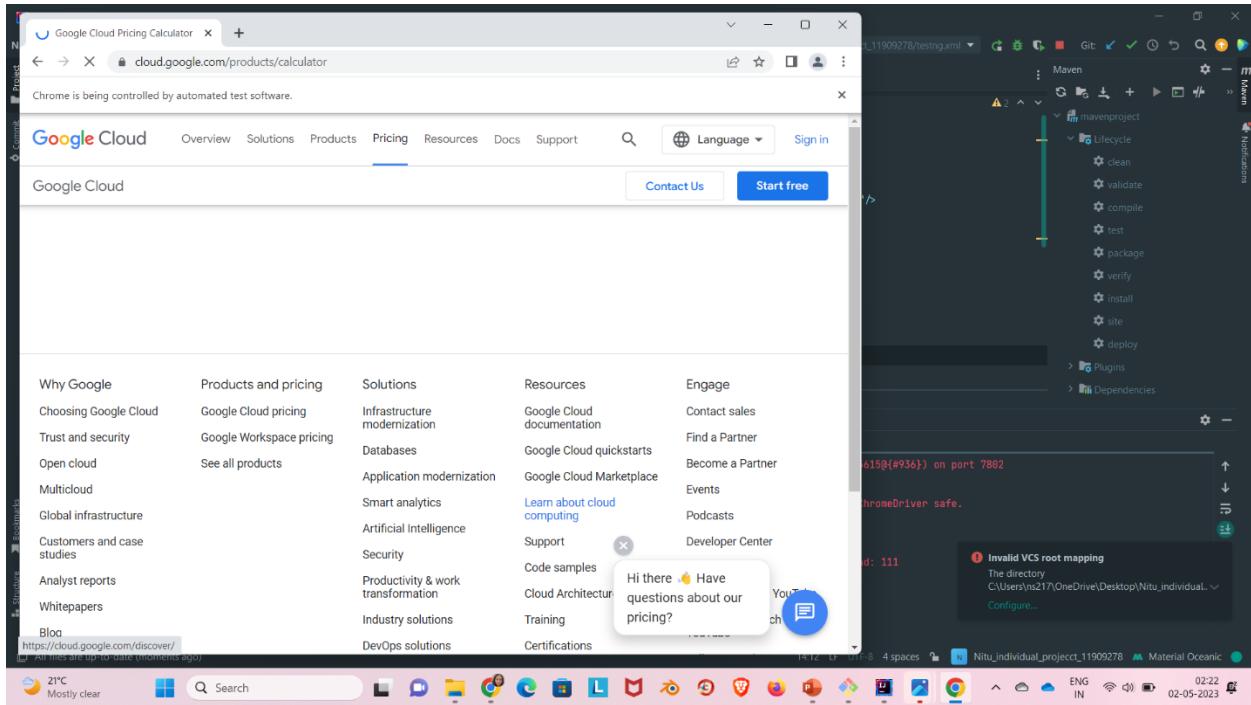
public void check_Data_is_correct()
{
    Assert.assertEquals(VM_OF_Class, expected: "Provisioning model: Regular");
    Assert.assertEquals(region, expected: "Region: Frankfurt");
    Assert.assertEquals(S_SD, expected: "Local SSD: 2x375 GiB\n" + "Committed Use Discount applied");
    Assert.assertEquals(No_of_Instance, expected: "Instance type: n1-standard-8\n" + "Committed Use Discount app");
    Assert.assertEquals(Time_Taken, expected: "Commitment term: 1 Year");
    Assert.assertEquals(Cost_of_USD, expected: "Total Estimated Cost: USD 1,081.20 per 1 month");
}
```

## Step 6 :- Using In Different Browser

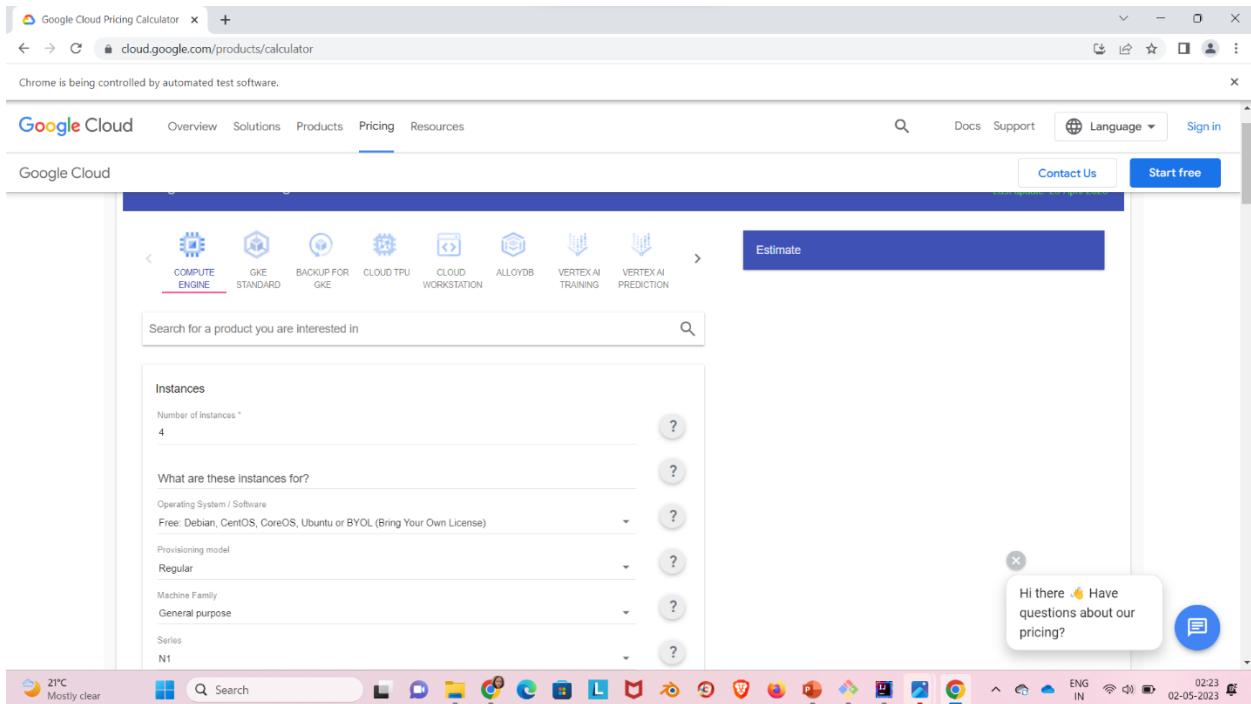
The screenshot shows the IntelliJ IDEA interface with the project 'Nitu\_individual\_projectt\_11909278' open. The code editor displays the file 'testing.xml'. This is a TestNG configuration file that defines two test suites: 'Test1' and 'Test2'. Each suite runs tests for the 'Google\_Price\_Cal\_Test' class, specifically the 'check\_Data\_is\_correct' method, using the 'chrome' and 'firefox' browsers respectively. The Maven tool window on the right shows the 'Lifecycle' section with several goals listed.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.8.dtd">
<suite name="All Test Suite">
    <test verbose="2" preserve-order="true" name="Test1">
        <parameter name="browser" value="chrome"/>
        <parameter name="Url" value="https://cloud.google.com/products/calculator"/>
        <classes>
            <class name="org.example.Google_Price_Cal_Test">
                <methods><include name="checkInformationInVmClassString"/>
                <include name="check_Data_is_correct"/>
            </methods>
        </class>
    </classes>
    </test>
    <test verbose="2" preserve-order="true" name="Test2">
        <parameter name="browser" value="firefox"/>
        <parameter name="Url" value="https://Cloud.google.com/products/calculator"/>
        <classes>
            <class name="org.example.Google_Price_Cal_Test">
                <methods><include name="checkInformationInVmClassString"/>
                <include name="check_Data_is_correct"/>
            </methods>
        </class>
    </classes>
    </test>
</suite>
```

## Step 7 :- Start Running In Chrome Browser



## Step 8 :- Start Running In Chorme Browser



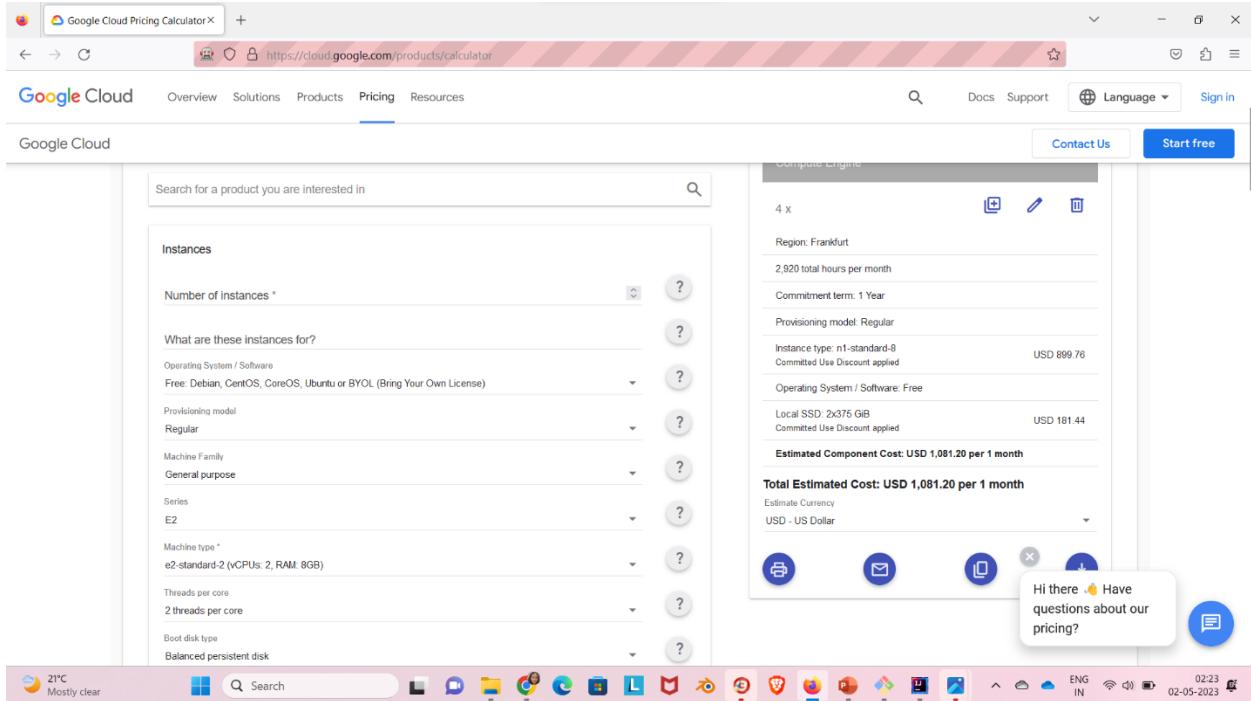
## Step 9 :- Start Running In Chrome Browser

The screenshot shows the Google Cloud Pricing Calculator interface in a Chrome browser window. The URL is [cloud.google.com/products/calculator](https://cloud.google.com/products/calculator). The page displays a form for configuring Compute Engine instances. On the left, there's a sidebar with various service icons. The main area has tabs for ENGINE, STANDARD, GKE, WORKSTATION, TRAINING, and PREDICTION. The ENGINE tab is selected. A search bar at the top says "Search for a product you are interested in". Below it, the "Instances" section includes fields for "Number of Instances" (set to 4), "What are these instances for?", "Operating System / Software" (Free: Debian, CentOS, CoreOS, Ubuntu or BYOL), "Provisioning model" (Regular), "Machine Family" (General purpose), "Series" (E2), "Machine type" (e2-standard-2), and "Threads per core" (2 threads per core). To the right, a detailed breakdown of costs is shown for 4x E2 instances in Frankfurt (Region). It lists the instance type (n1-standard-8), operating system (Free), local SSD (2x375 GiB), and committed use discount applied. The estimated component cost is USD 1,081.20 per month. A total estimated cost of USD 1,081.20 per month is also displayed. A tooltip message "Hi there! Have questions about our pricing?" appears near the bottom right.

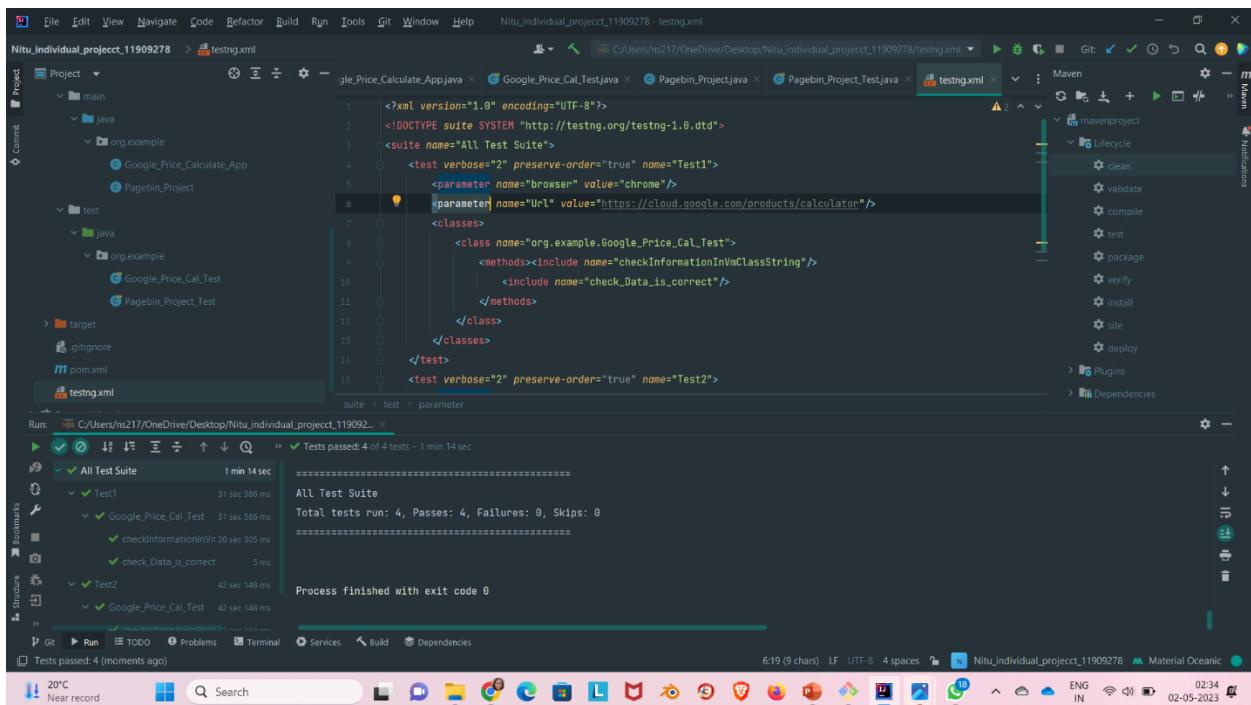
## Step 10 :- Start Running In FireFox Browser

The screenshot shows the Google Cloud Pricing Calculator interface in a FireFox browser window. The URL is [cloud.google.com/products/calculator](https://cloud.google.com/products/calculator). The page layout is identical to the Chrome version, with the ENGINE tab selected. The configuration for 4x E2 instances in Frankfurt is identical to the previous screenshot. The detailed cost breakdown and total estimated cost are also the same. A tooltip message "Hi there! Have questions about our pricing?" is visible near the bottom right.

## Step 11 :- Start Running In FireFox Browser



## Step 12 :- Build Successful



## Step 13 :- Build Successful

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Nitu\_individual\_projectt\_11909278". It contains a "main" module with a "java" directory containing "org.example" packages for "Google\_Price\_Calculate\_App" and "Pagebin\_Project". A "test" module also has a "java" directory with "org.example" packages for "Google\_Price\_Cal\_Test" and "Pagebin\_Project\_Test".
- Maven Tool Window:** On the right, the Maven tool window shows the "Lifecycle" section with "clean", "validate", "compile", "test", "package", "verify", "install", "site", and "deploy" goals.
- Run Tab:** The "Run" tab shows a successful build for "mavenproject [test]". The output log shows:

```
[INFO] [INFO] BUILD SUCCESS
[INFO] ...
[INFO] Total time: 01:26 min
[INFO] Finished at: 2023-05-02T02:39:39+05:30
[INFO] ...
```

Process finished with exit code 0
- System Tray:** The system tray at the bottom indicates it's 20°C, mostly clear, and shows the date and time as 02-05-2023.

## Step 14 :- All Test Cases Pass

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Nitu\_individual\_projectt\_11909278". It contains "encodings.xml", "jarRepositories.xml", "misc.xml", "uiDesigner.xml", "vcs.xml", and "workspace.xml".
- Maven Tool Window:** The Maven tool window shows the "Lifecycle" section with "clean", "validate", "compile", "test", "package", "verify", "install", "site", and "deploy" goals.
- Run Tab:** The "Run" tab shows a successful build for "mavenproject\_Nitu\_Singh\_11909278 [test]". The output log shows:

```
Test2
Tests run: 2, Failures: 0, Skips: 0
=====
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 74.099 s - in TestSuite
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 4, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:28 min
[INFO] Finished at: 2023-05-02T04:26:44+05:30
[INFO] ...
```

Process finished with exit code 0
- System Tray:** The system tray at the bottom indicates it's 20°C, mostly clear, and shows the date and time as 02-05-2023.

## Step 15 :- Target-Surefire-report-index.html

localhost:63342/Nitu\_individual\_projectt\_11909278/mavenproject/target/surefire-reports/index.html?\_jtt=evqh5bh2onb5eu9rf6cvj1vc0&\_ij\_.reload=RELOAD\_ON\_SAVE#

20°C Mostly clear Search ENG IN 02:45 02-05-2023

## Step 16 :- Target-Surefire-report-emailable-report.html

Test	# Passed	# Skipped	# Retried	# Failed	Time (ms)	Included Groups	Excluded Groups
All Test Suite							
Test1	2	0	0	0	35,512		
Test2	2	0	0	0	41,986		
Total	4	0	0	0	77,498		

Class	Method	Start	Time (ms)
All Test Suite			
Test1 — passed			
org.example.Google_Price_Cal_Test	checkInformationInVmClassString	1682975547629	19934
	check_Data_is_correct	1682975557587	4
Test2 — passed			
org.example.Google_Price_Cal_Test	checkInformationInVmClassString	16829755584723	23186
	check_Data_is_correct	1682975607911	0

Test1

org.example.Google\_Price\_Cal\_Test#checkInformationInVmClassString

[back to summary](#)

org.example.Google\_Price\_Cal\_Test#check\_Data\_is\_correct

[back to summary](#)

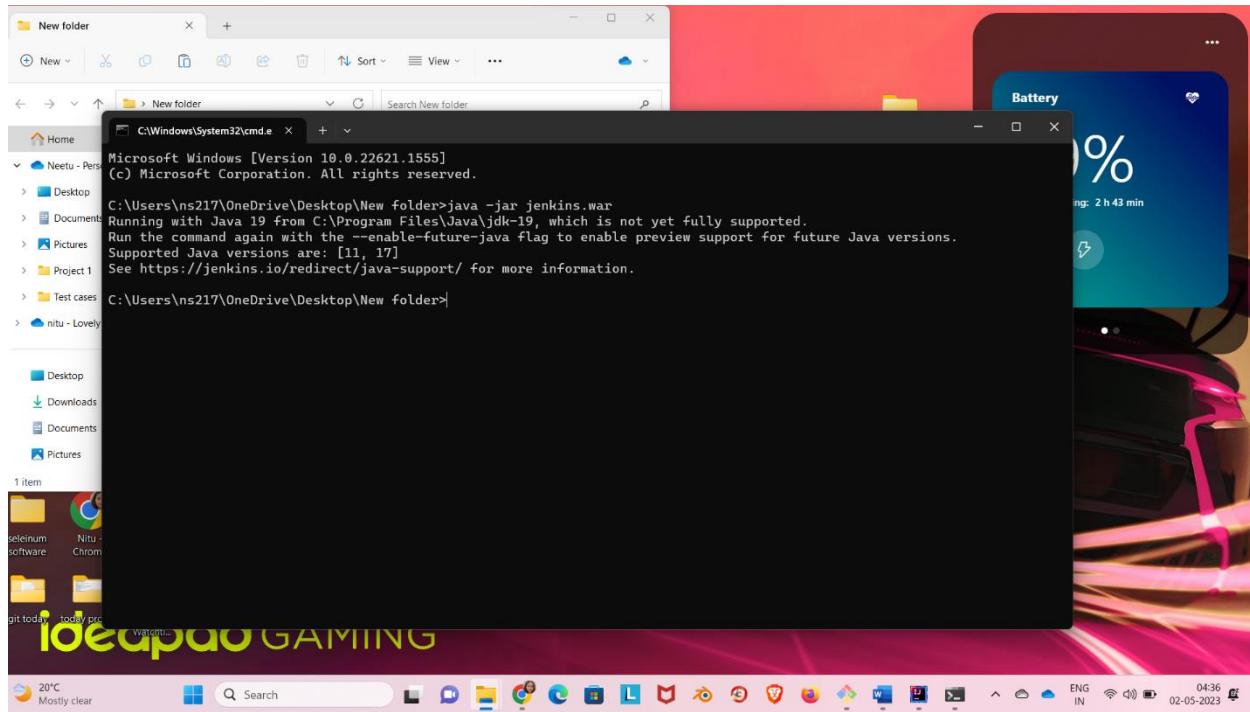
Test2

org.example.Google\_Price\_Cal\_Test#checkInformationInVmClassString

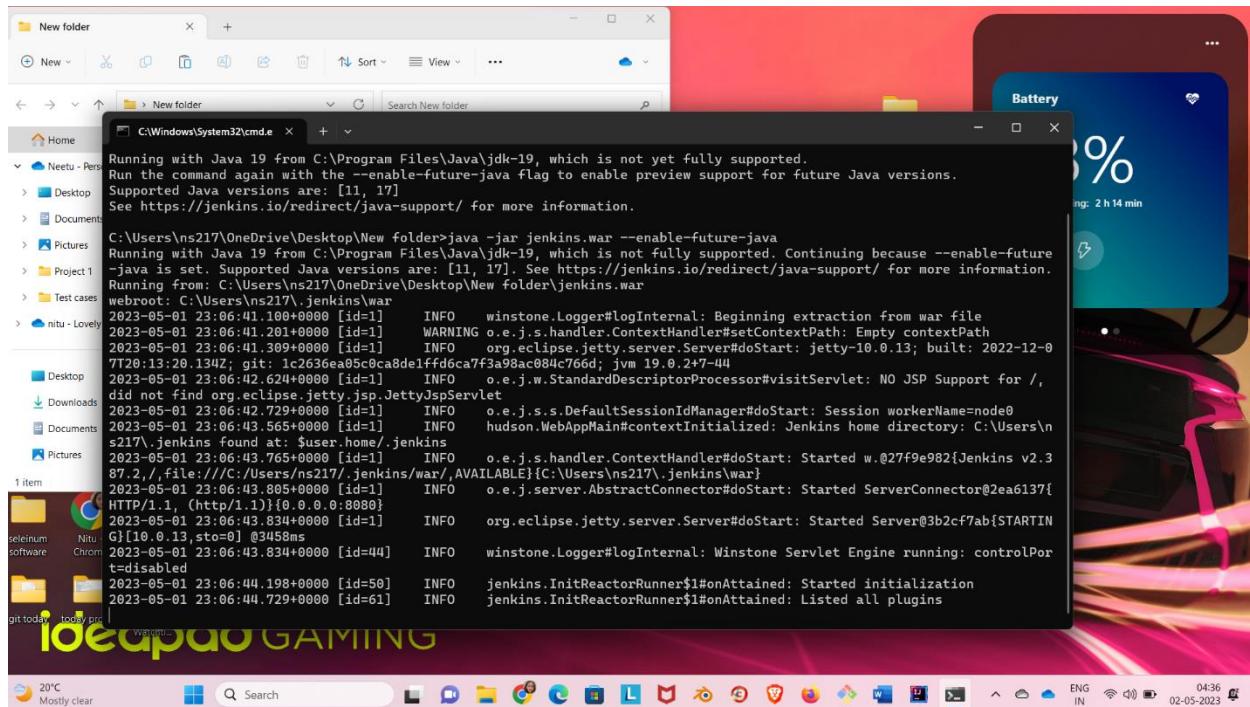
[back to summary](#)

20°C Mostly clear Search ENG IN 02:45 02-05-2023

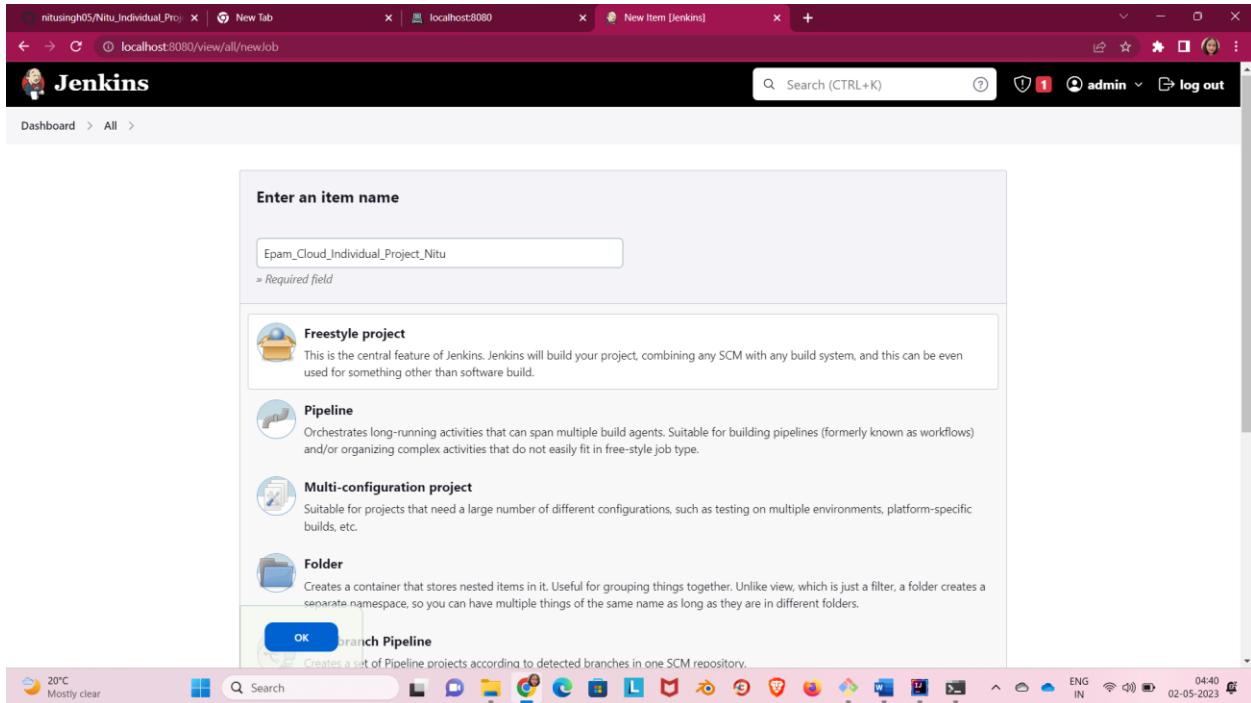
## Step 17 :- Start Jenkins Running



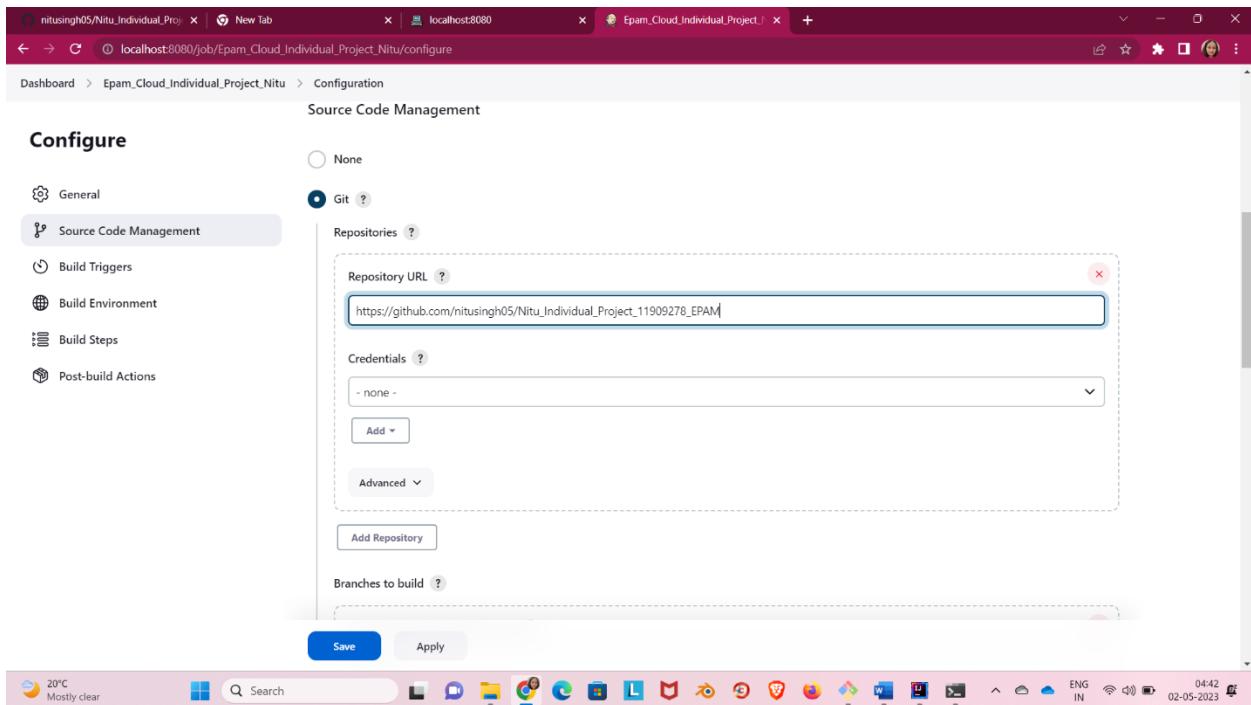
## Step 18 :- Running Jenkins



## Step 19 :- Start Jenkins Running Item Name



## Step 20 :- Start Jenkins Configuration git repository URL



## Step 21 :- Start Jenkins Configuration- Source code management

The screenshot shows the Jenkins configuration interface for a project named 'Epam\_Cloud\_Individual\_Project\_Nitu'. The 'Source Code Management' section is selected. Under 'Branches to build', the 'Branch Specifier (blank for 'any')' field contains the value '/master'. The 'Repository browser' dropdown is set to '(Auto)'. The 'Additional Behaviours' section has an 'Add' button. At the bottom, there are 'Save' and 'Apply' buttons.

## Step 22 :- start Jenkins Configuration Build Triggers

The screenshot shows the Jenkins configuration interface for 'Build Triggers'. The 'Build periodically' option is selected with a schedule of '\*/5 \* \* \* \*'. A warning message states: '⚠ Spread load evenly by using 'H/5 \* \* \* \*' rather than '\*/5 \* \* \* \*''. Below this, there are options for 'GitHub hook trigger for GITScm polling' and 'Poll SCM'. The 'Build Environment' section includes options for 'Delete workspace before build starts' and 'Use secret text(s) or file(s)'. At the bottom, there are 'Save' and 'Apply' buttons.

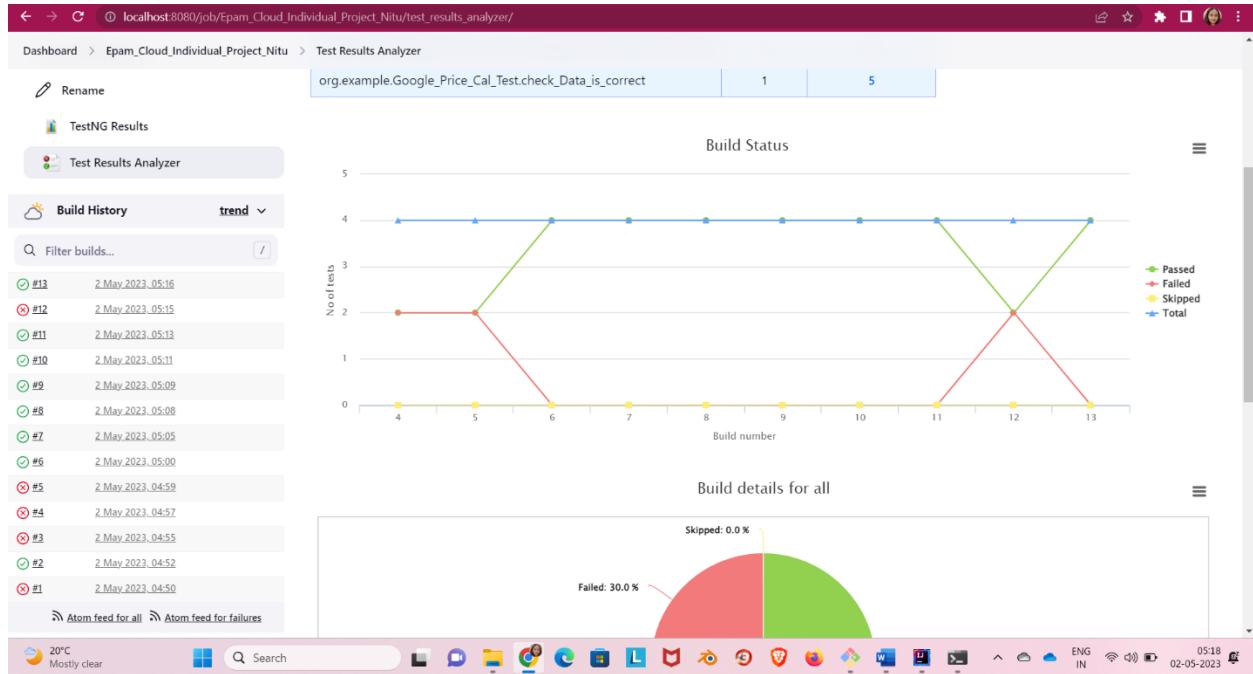
## Step 23 :- start Jenkins Configuration-Build Steps

The screenshot shows the Jenkins configuration interface for a job named 'Epam\_Cloud\_Individual\_Project\_Nitu'. The 'Build Steps' section is active, displaying a step titled 'Invoke top-level Maven targets'. The 'Goals' field contains 'clean test'. Below this, there is an 'Advanced' dropdown and a 'Save' button.

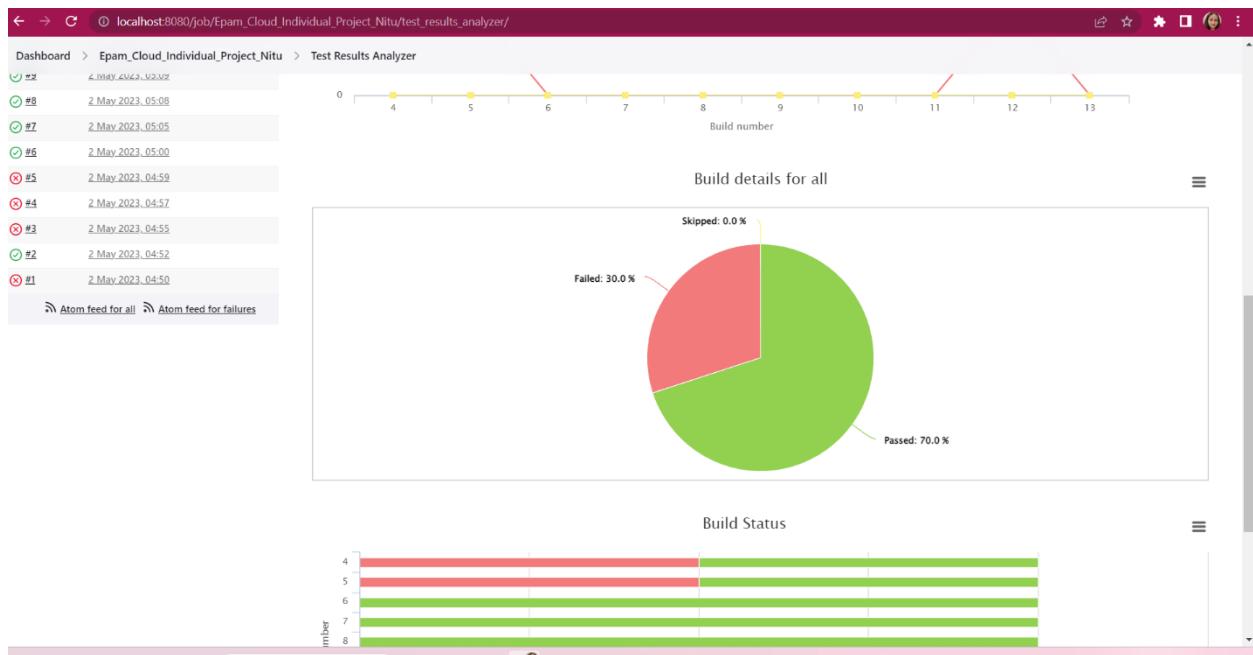
## Step 24 :- Test Cases Report In Jenkins

The screenshot shows the Jenkins Test Results Analyzer page. It displays a summary table for 'org.example' with 70% (85%) passed, 3 transitions, and a trend chart showing 13 builds. Below this is a table for 'Top 10 Most Broken Tests' showing two entries: 'org.example.Google\_Price\_Cal\_Test.checkInformationInVmClassString' and 'org.example.Google\_Price\_Cal\_Test.check\_Data\_is\_correct', both with 1 failure and 5 recent failed builds. A 'Build Status' trend chart at the bottom shows the number of tests over time, with a green line for 'Passed' and a red line for 'Failed'.

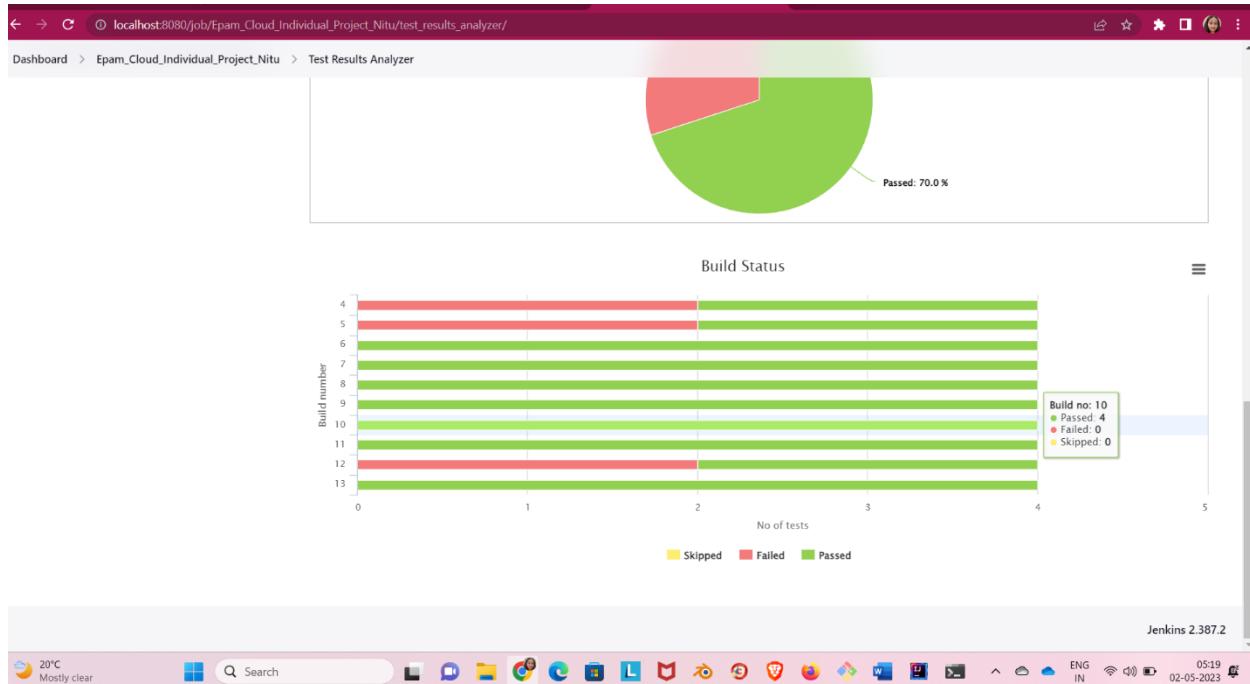
## Step 25 :- Test Cases Report In Jenkins



## Step 26 :- Test Cases Report In Jenkins



## Step 27 :- Test Cases Report In Jenkins



## Step 28 :- Test Cases Report In Jenkins

