| **Name of Student:** Ajay Karthikesan | | | |
| --- | --- | --- | --- |
| **Roll Number:** 57 | | **Practical Number:** 5 | |
| **Aim of Practical:**  1. Installation of TestNg , running testNg and TestNg annotations  2. Demonstrate data driven Framework.  3. Demonstrate Validation testing  4. Perform regression testing | | | |
| **DOP:** 16.10.23 | | **DOS:** 2.11.23 | |
| **CO Mapped:** - | **PO Mapped:** - | **Faculty Signature:** | **Marks:** |

## 

## Practical No. 5

**Aim:**

1. Installation of TestNg , running testNg and TestNg annotations

2. Demonstrate data driven Framework.

3. Demonstrate Validation testing

4. Perform regression testing

**Theory:**

**TestNG** is a testing framework for the Java programming language created by Cédric Beust and inspired by JUnit and NUnit. The design goal of TestNG is to cover a wider range of test categories: unit, functional, end-to-end, integration, etc., with more powerful and easy-to-use functionalities.

TestNG's main features include:

Annotation support.

Support for data-driven/parameterized testing (with @DataProvider and/or XML configuration).

Support for multiple instances of the same test class (with @Factory)

Flexible execution model. TestNG can be run either by Ant via build.xml (with or without a test suite defined), or by an IDE plugin with visual results. There isn't a TestSuite class, while test suites, groups and tests selected to run are defined and configured by XML files.

Concurrent testing: run tests in arbitrarily big thread pools with various policies available (all methods in their own thread, one thread per test class, etc.), and test whether the code is multithread safe.

Embeds BeanShell for further flexibility.

Default JDK functions for runtime and logging (no dependencies).

Dependent methods for application server testing.

Distributed testing: allows distribution of tests on slave machines.

**Data Driven framework** is used to drive test cases and suites from an external data feed. The data feed can be data sheets like xls, xlsx, and csv files. A Data Driven Framework in Selenium is a technique of separating the “data set” from the actual “test case” (code).

**Validation testing** is a process that ensures a new software product meets consumer needs. It helps development teams ensure that the work meets stakeholders' expectations.

What is validation testing?

Validation testing ensures that a product meets the client's needs.

It helps development teams ensure that the work meets stakeholders' expectations.

It gives development teams a final chance to address any defects or mismatches between the application and the requirements.

How is validation testing performed?

Product development teams might perform validation testing to learn about the integrity of the product itself and its performance in different environments.

Validation testing includes testing like functional testing, system testing, integration, and user acceptance testing.

Validation testing is also known as dynamic testing.

**Regression Testing** is a type of testing in the software development cycle that runs after every change to ensure that the change introduces no unintended breaks. Regression testing addresses a common issue that developers face — the emergence of old bugs with the introduction of new changes.

**Code:**

File: NewTest.java

package vesit.ajayk57.dscc.stqa\_practical5;

import static org.testng.Assert.assertEquals;

import java.time.Duration;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.BeforeMethod;

import org.testng.annotations.Test;

public class NewTest {

@Test

public void f() {

System.out.println("f()");

assert(true);

}

@Test

public void eightComponents() {

WebDriver driver = new ChromeDriver();

driver.get("https://www.selenium.dev/selenium/web/web-form.html");

String title = driver.getTitle();

assertEquals("Web form", title);

driver.manage().timeouts().implicitlyWait(Duration.ofMillis(500));

WebElement textBox = driver.findElement(By.name("my-text"));

WebElement submitButton = driver.findElement(By.cssSelector("button"));

textBox.sendKeys("Selenium");

submitButton.click();

WebElement message = driver.findElement(By.id("message"));

String value = message.getText();

assertEquals("Received!", value);

driver.quit();

}

@BeforeMethod

public void beforeMethod() {

System.out.println("Runs before the test methods");

}

@AfterMethod

public void afterMethod() {

System.out.println("Runs after the test methods");

}

}

File: LoginTest.java

package vesit.ajayk57.dscc.stqa\_practical5;

import java.io.IOException;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class LoginTest {

private WebDriver driver;

private final String USER\_DATA\_PATH\_STR = "users.xlsx";

private final int SHEET\_NUMBER = 0;

private ReadExcelFile readExcelFile;

@BeforeTest

public void setupTest() throws IOException {

driver = new ChromeDriver();

readExcelFile = new ReadExcelFile(USER\_DATA\_PATH\_STR, SHEET\_NUMBER);

}

@Test(dataProvider = "testdata")

public void simpleLogin(String username, String password) throws InterruptedException {

driver.get("https://www.saucedemo.com");

driver.findElement(By.id("user-name")).sendKeys(username);

driver.findElement(By.id("password")).sendKeys(password);

driver.findElement(By.id("login-button")).click();

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

Assert.assertTrue(driver.getCurrentUrl().matches("https://www.saucedemo.com/inventory.html"),

"Invalid credentials");

}

@AfterTest

void ProgramTermination() {

driver.quit();

}

@DataProvider(name = "testdata")

public Object[][] testDataExample() throws IOException {

final int numRows = readExcelFile.getRowCount();

Object[][] loginCredentials = new Object[numRows][2];

for (int i = 0; i < numRows; i++) {

loginCredentials[i][0] = readExcelFile.getData(i, 0);

loginCredentials[i][1] = readExcelFile.getData(i, 1);

}

return loginCredentials;

}

}

File: ReadExcelFile.java

package vesit.ajayk57.dscc.stqa\_practical5;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

public class ReadExcelFile {

final XSSFWorkbook workBook;

final XSSFSheet sheet;

public ReadExcelFile(final String excelfilePathStr, final int sheetNum) throws IOException {

final File excelFile = new File(excelfilePathStr);

final FileInputStream stream = new FileInputStream(excelFile);

workBook = new XSSFWorkbook(stream);

sheet = workBook.getSheetAt(sheetNum);

}

public String getData(int row, int column) {

return sheet.getRow(row).getCell(column).getStringCellValue();

}

public int getRowCount() {

return sheet.getPhysicalNumberOfRows() - 1;

}

}

File: ValidationTest.java

package vesit.ajayk57.dscc.stqa\_practical5;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.Assert;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

import org.testng.annotations.Test;

public class ValidationTest {

private WebDriver driver;

private final String websiteURL = "https://agilemanifesto.org/";

@BeforeTest

public void setupTest() {

driver = new ChromeDriver();

driver.get(websiteURL);

}

@Test

public void testPageTitle() {

final String expectedTitle = "Manifesto for Agile Software Development";

final String actualTitle = driver.getTitle();

Assert.assertEquals(actualTitle, expectedTitle, "Page title doesn't match the expected title.");

}

@AfterTest

public void teardown() {

driver.quit();

}

}

File: BaseTest.java

package vesit.ajayk57.dscc.stqa\_practical5.regression;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.testng.annotations.AfterTest;

import org.testng.annotations.BeforeTest;

public class BaseTest {

final protected String WEBSITE\_URL = "https://www.saucedemo.com/";

protected WebDriver driver;

@BeforeTest

public void setup() {

driver = new ChromeDriver();

driver.get("https://www.saucedemo.com/");

}

@AfterTest

public void teardown() {

if (driver != null) {

driver.quit();

}

}

}

File: LoginTest.java

package vesit.ajayk57.dscc.stqa\_practical5.regression;

import java.time.Duration;

import org.openqa.selenium.By;

import org.openqa.selenium.WebElement;

import org.testng.Assert;

import org.testng.annotations.Test;

public class LoginTest extends BaseTest {

@Test(priority = 1)

public void testLoginWithValidCredentials() {

login("standard\_user", "secret\_sauce");

Assert.assertTrue(isLoggedIn(), "Login with valid credentials failed.");

}

@Test(priority = 2)

public void testLoginWithInvalidCredentials() {

login("locked\_out\_user", "secret\_sauce");

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));

Assert.assertTrue(isErrorMessageDisplayed(), "Error message not displayed for invalid login.");

}

private void login(String username, String password) {

driver.get("https://www.saucedemo.com/");

driver.findElement(By.id("user-name")).sendKeys(username);

driver.findElement(By.id("password")).sendKeys(password);

driver.findElement(By.id("login-button")).click();

}

private boolean isLoggedIn() {

WebElement cartIcon = driver.findElement(By.id("shopping\_cart\_container"));

return cartIcon.isDisplayed();

}

private boolean isErrorMessageDisplayed() {

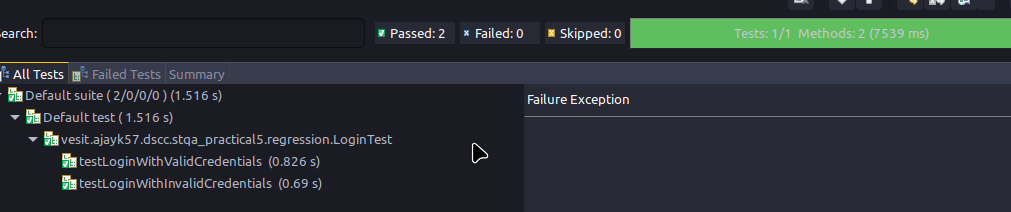
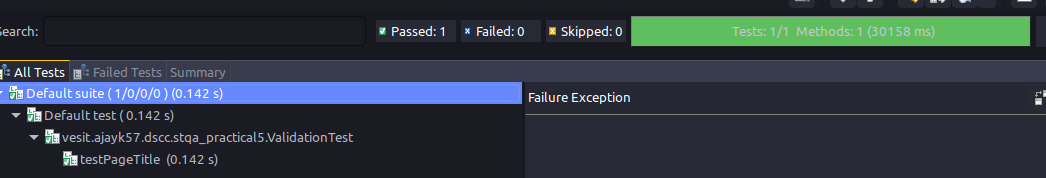
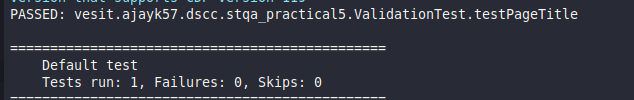
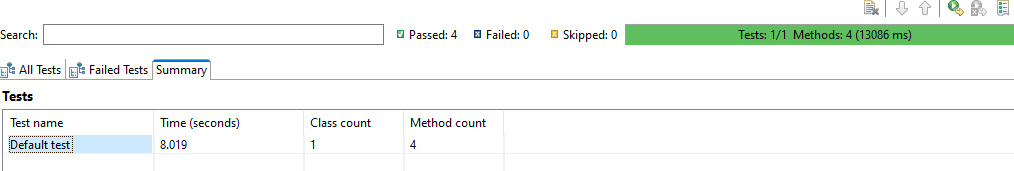
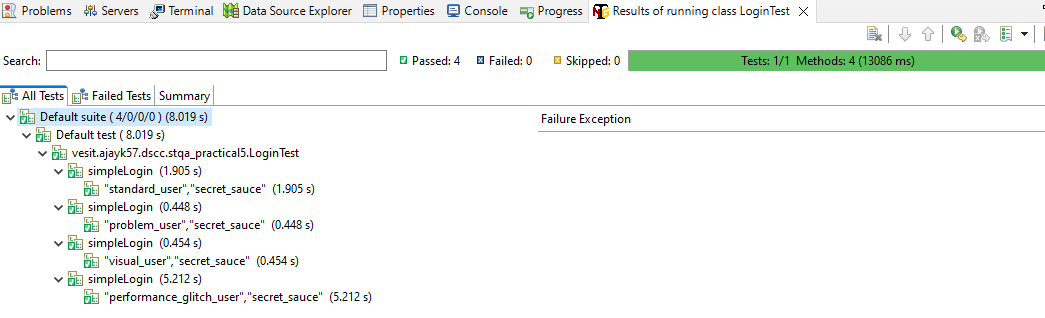
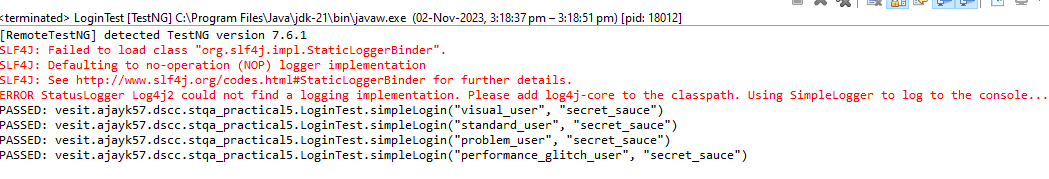
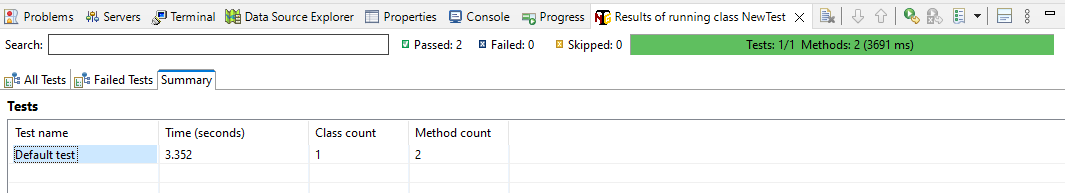
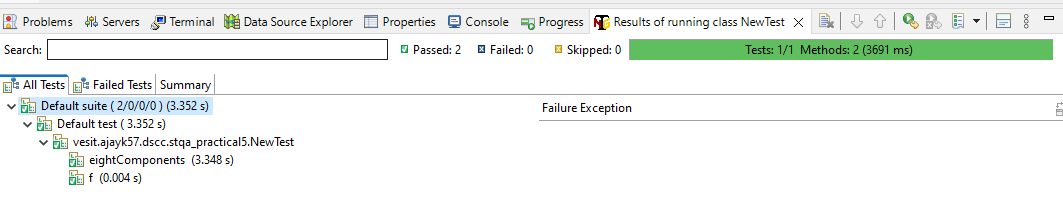
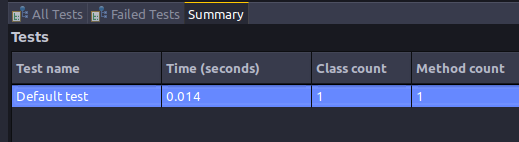
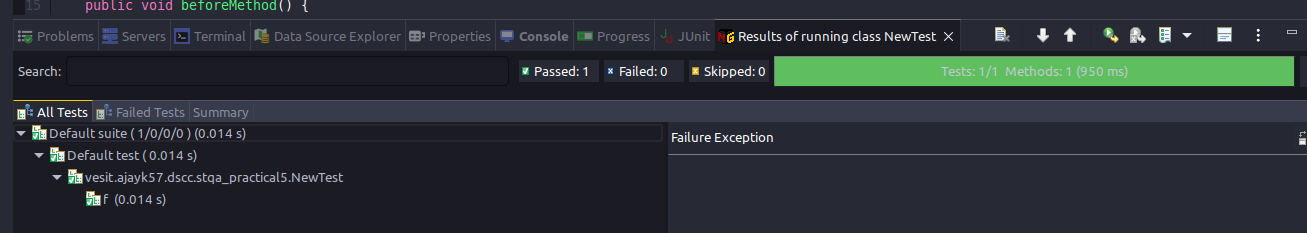
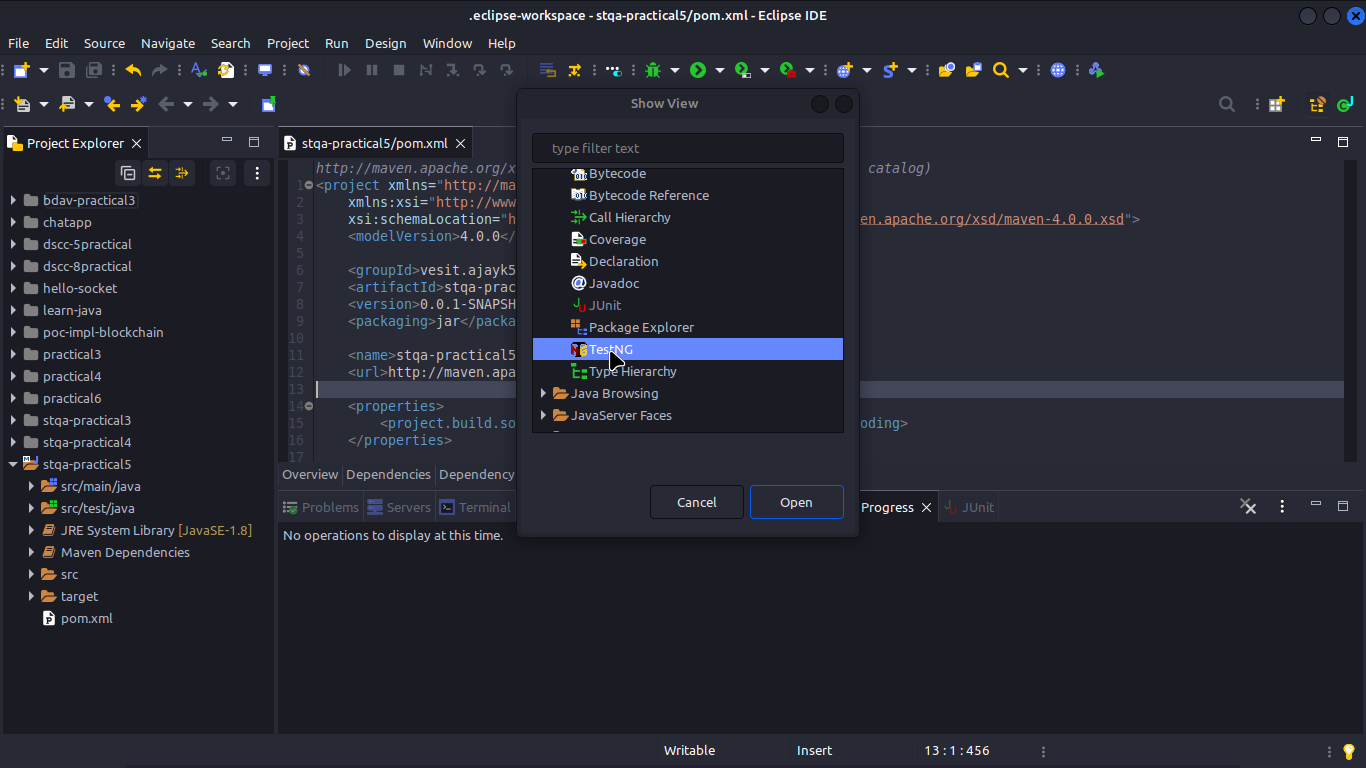
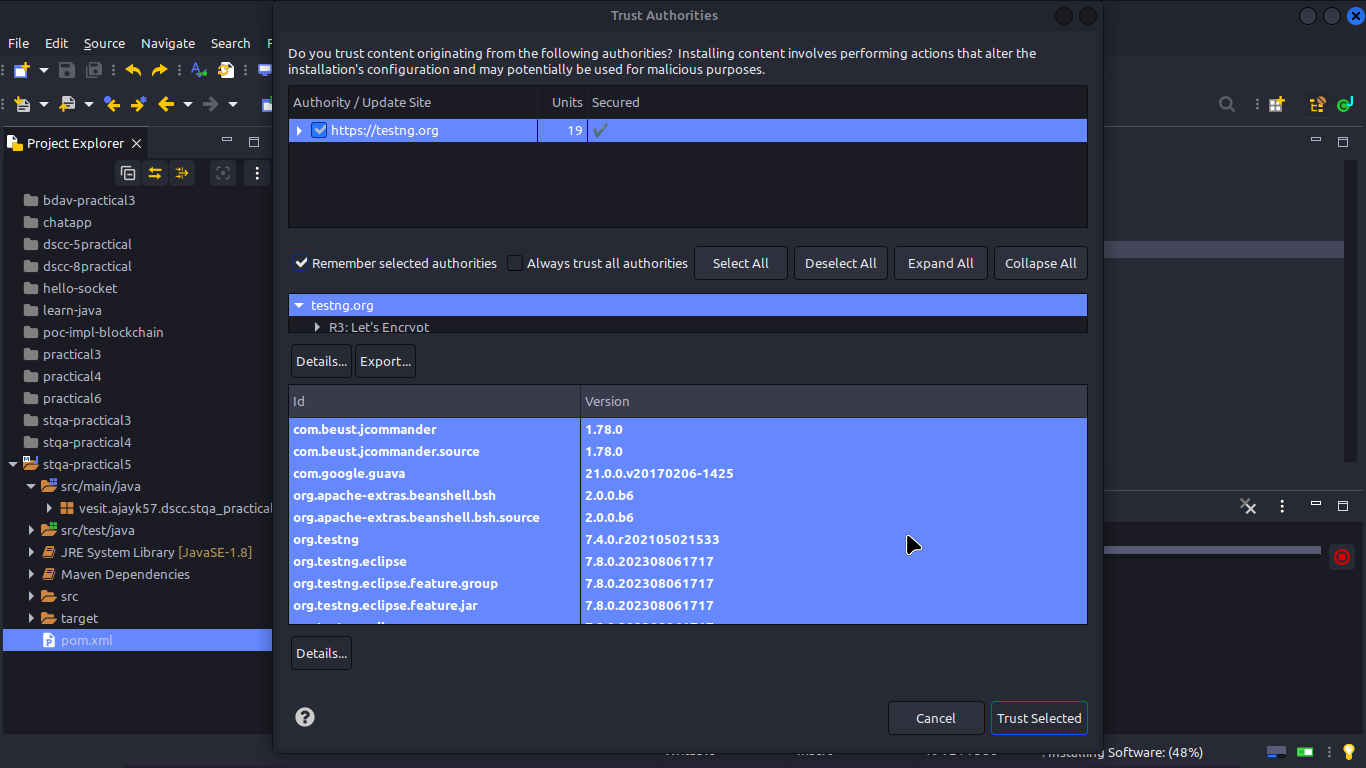
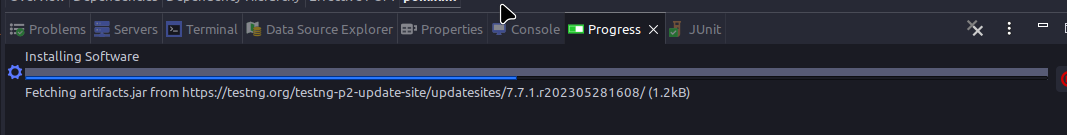
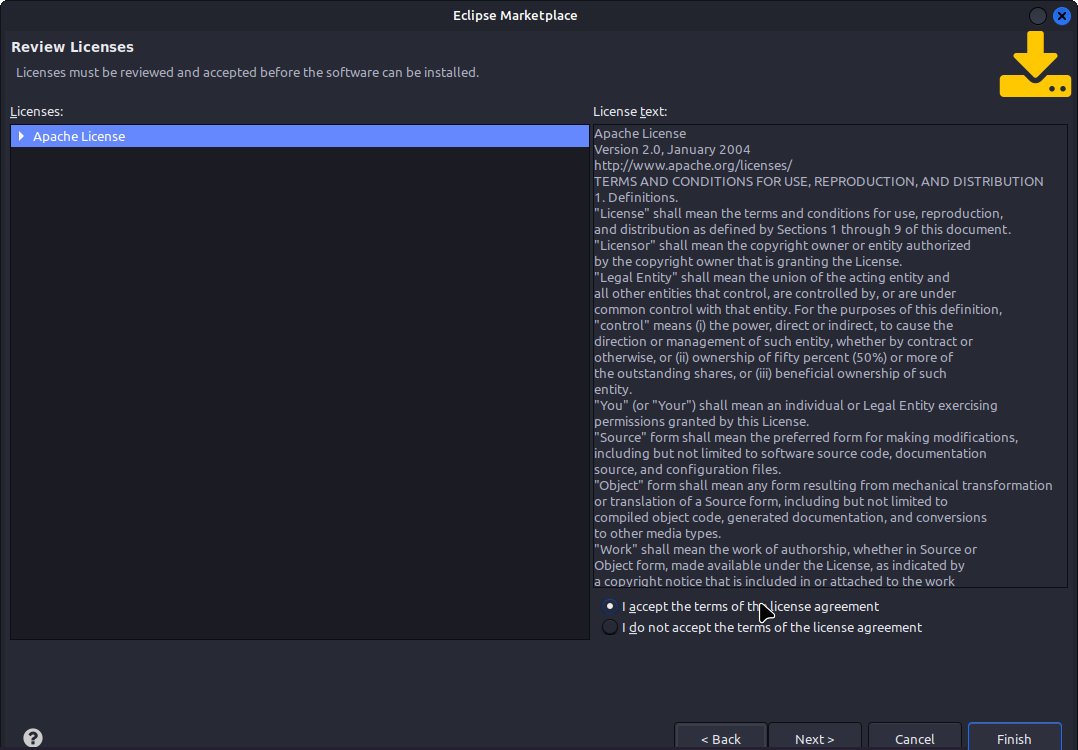
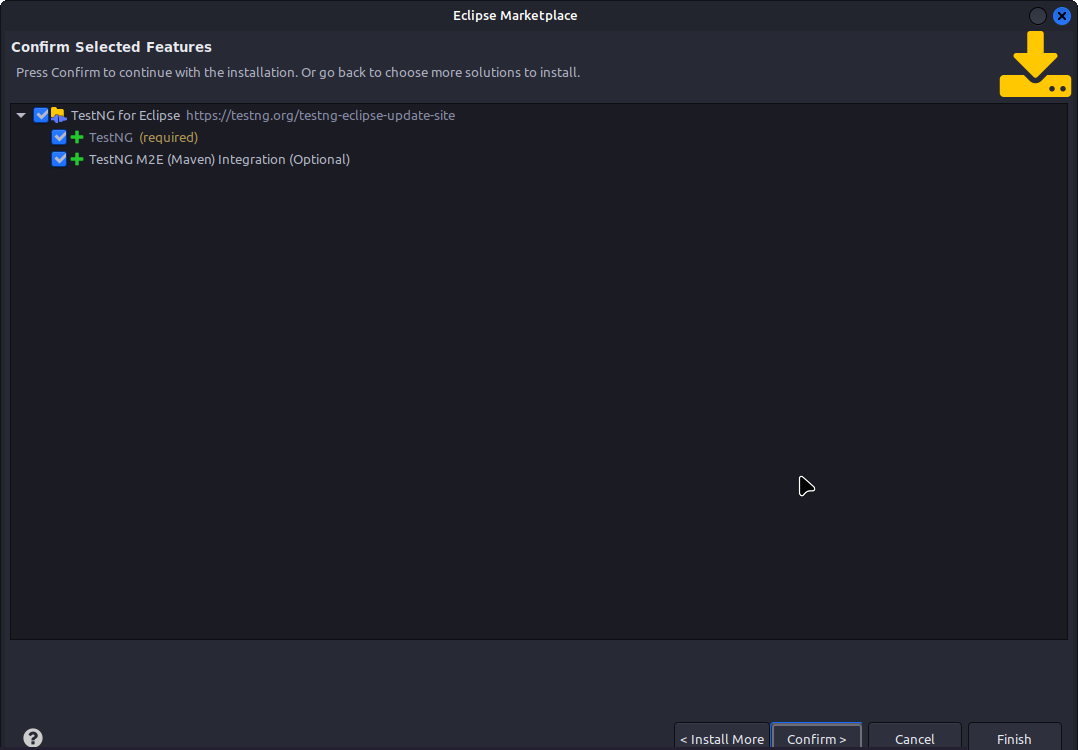
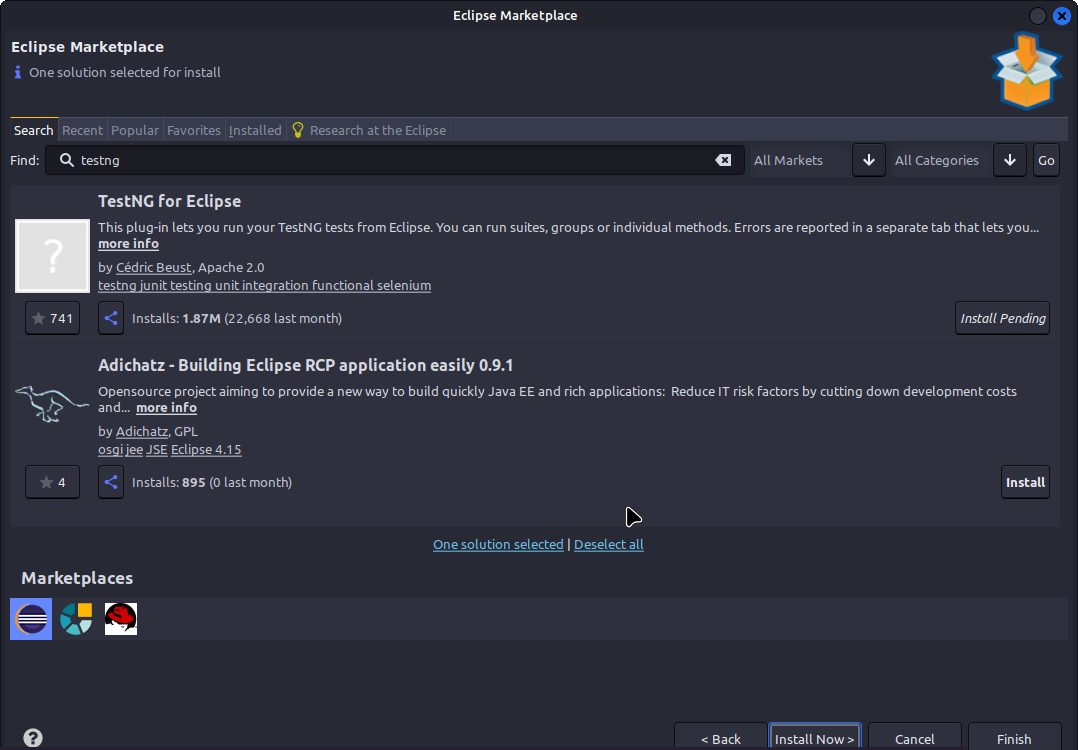
WebElement errorElement = driver.findElement(By.cssSelector("[data-test='error']"));

return errorElement.isDisplayed();

}

}

**Output:**



**Conclusion:**

I learnt how to…

1. Installation of TestNg , running testNg and TestNg annotations

2. Demonstrate data driven Framework.

3. Demonstrate Validation testing

4. Perform regression testing