**Ex No: 5a** **Build a data-driven framework using Selenium and TestNG**

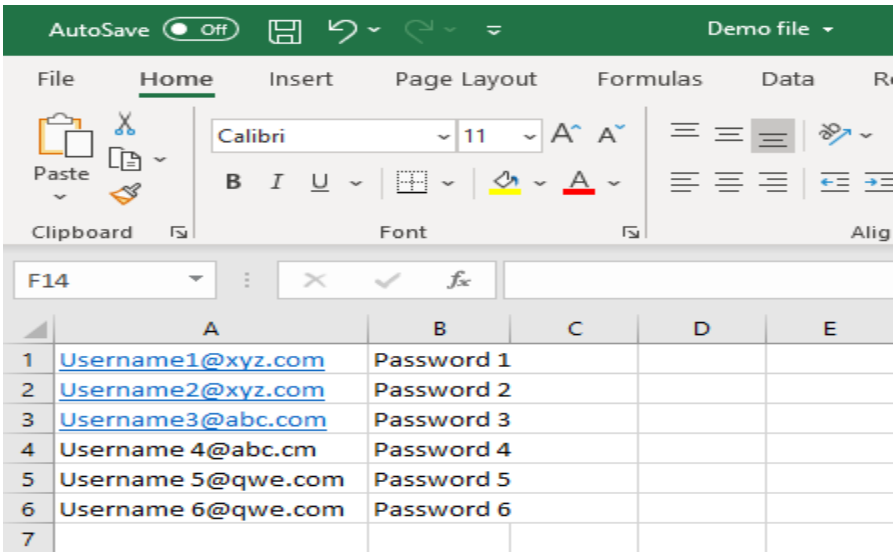
**Date:**

**Scope:**

To build a data-driven framework using Selenium and TestNG

**Procedure:**

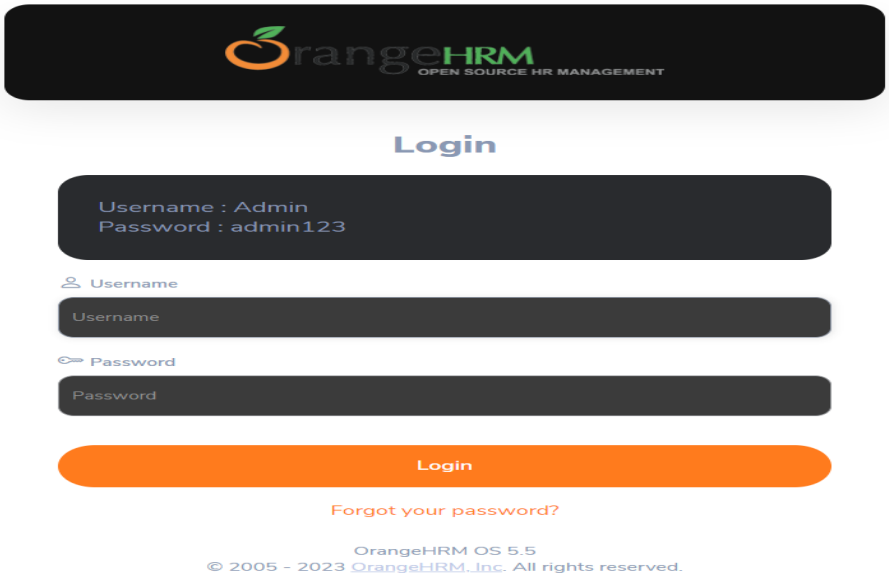
Step 1:Create a excel sheet which has username and password in individual columns.




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Username1@xyz.com | Password 1 | | | |
| 2 | Username2@xyz.com | Password 2 | | | |
| 3 | Username3@abc.com | Password 3 | | | |
| 4 | Username 4@abc.cm | Password 4 | | | |
| 5 | Username 5@qwe.com | Password 5 | | | |
| 6 | Username 6@qwe.com | Password 6 | | | |
| 7 | | | | | |

Step 2:

Here the targeted website is https://opensource-demo.orangehrmlive.com/



Step 3:

Go to the Eclipse IDE and create a project. Add all the dependencies for TestNG, Selenium and Apache POI.

Step 4:

Create a code to write the functionality

Code:

```
import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.AfterMethod;

import org.testng.annotations.DataProvider;

import org.testng.annotations.Test;

public class ExcelExample{

@Test(dataProvider="testdata")

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

System.setProperty("webdriver.chrome.driver", "Path of Chrome Driver");

Webdriver driver = new ChromeDriver();

driver.get("<a href="https://opensource-demo.orangehrmlive.com/</a>");

driver.findElement(By.name("username")).sendKeys(username);

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

driver.findElement(By.name("commit")).click();

Thread.sleep(5000);

Assert.assertTrue(driver.getTitle().matches("BrowserStack Login | Sign Into The Best Mobile & Browser Testing Tool"), "Invalid credentials");

System.out.println("Login successful");

}

@AfterMethod

void ProgramTermination() {

driver.quit();
```

```
}
@DataProvider(name="testdata")
public Object[][] testDataExample(){
ReadExcelFile configuration = new ReadExcelFile("Path_of_Your_Excel_File");
int rows = configuration.getRowCount(0);
Object[][]signin_credentials = new Object[rows][2];

for(int i=0;i<rows;i++)
{
signin_credentials[i][0] = config.getData(0, i, 0);
signin_credentials[i][1] = config.getData(0, i, 1);
}
return signin_credentials;
}
}
```

In the above code, there is a “TestDataExample() method” in which the user has created an object instance of another class named “ReadExcelFile”. The user has mentioned the path to the excel file. The user has further defined a for loop to retrieve the text from the excel workbook. But to fetch the data from the excel file, one needs to write a class file for the same.

Code:

```
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;
}
```

**Result:**

Thus the data driven model has been build using Selenium and TestNG

**Ex No: 5 b Build Page object Model using Selenium and TestNG**

**Date:**

**Scope:**

To build Page object Model using Selenium and TestNG

**Procedure:**

Step 1: Download and Install Java

Cucumber and Selenium need Java to be installed on the system to run the tests.

Step 2 : Setup Maven

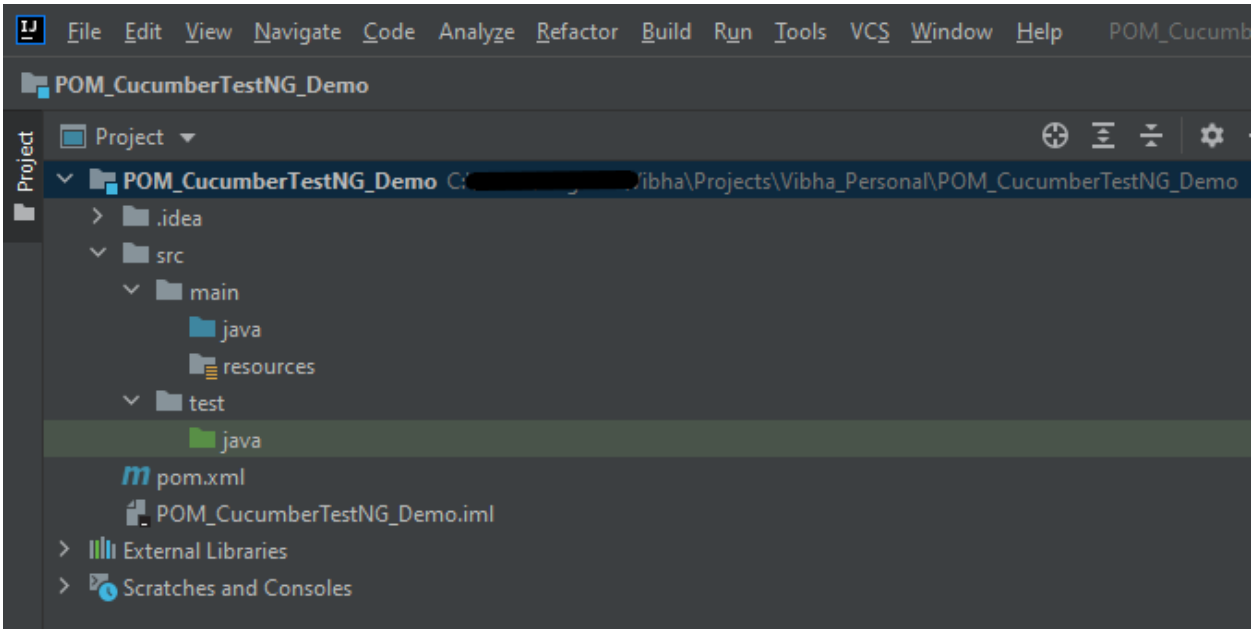
To build a test framework, we need to add a number of dependencies to the project.

Step 3 :Install Cucumber Eclipse Plugin (Only for Eclipse)

The cucumber plugin is an Eclipse plugin that allows eclipse to understand the Gherkin syntax. When we are working with cucumber we will write the feature files that contain Feature, Scenario, Given, When, Then, And, But, Tags, Scenario Outline, and Examples. By default, eclipse doesn't understand these keywords so it doesn't show any syntax highlighter. Cucumber Eclipse Plugin highlights the keywords present in Feature File. Refer to this tutorial to get more detail

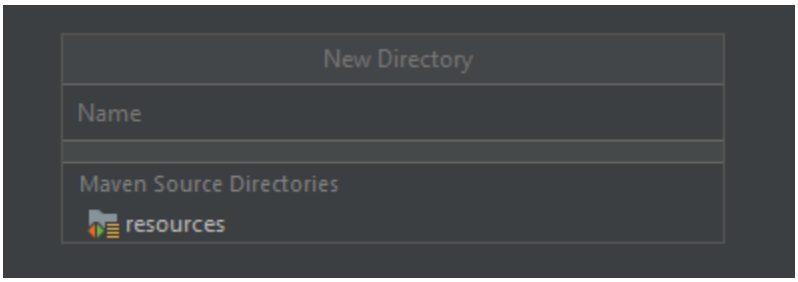
Step 4 : Create a new Maven Project

To create a new Maven project, go to the File -> New Project-> Maven-> Maven project-> Next -> Enter Group ID & Artifact ID -> Finish.

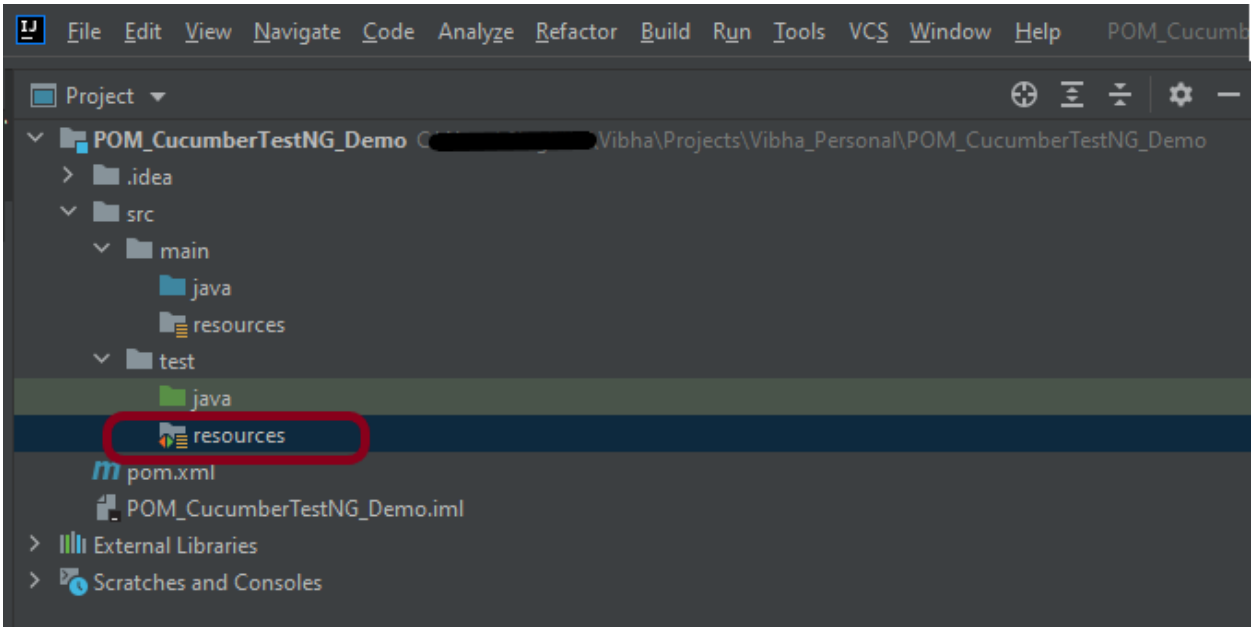


Step 5 – Create source folder src/test/resources to create test scenarios in the Feature file

A new Maven Project is created with 2 folders – src/main/java and src/test/java. To create test scenarios, we need a new source folder called – src/test/resources. To create this folder, right-click on test directory ->select New ->Directory, and then it shows Maven Source Directories as resources as shown below.




Double-click on the resources directory and a new source directory under your new Maven project is created as shown in the below image.




Step 6 – Add Selenium, TestNG, and Cucumber dependencies to the project

Add below mentioned Selenium, TestNG, and Cucumber dependencies to the project. I have added WebDriverManager dependency to the POM.xml to download the driver binaries automatically.

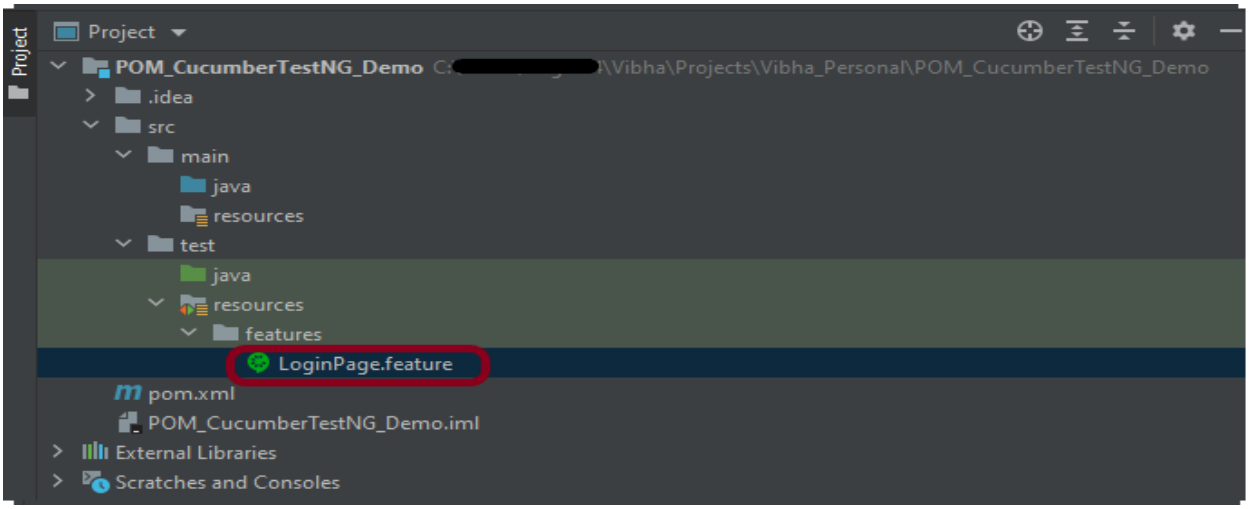
Step 7 – Add Maven Compiler Plugin and Surefire Plugin

The compiler plugin is used to compile the source code of a Maven project. This plugin has two goals, which are already bound to specific phases of the default lifecycle:

compile – compile main source files

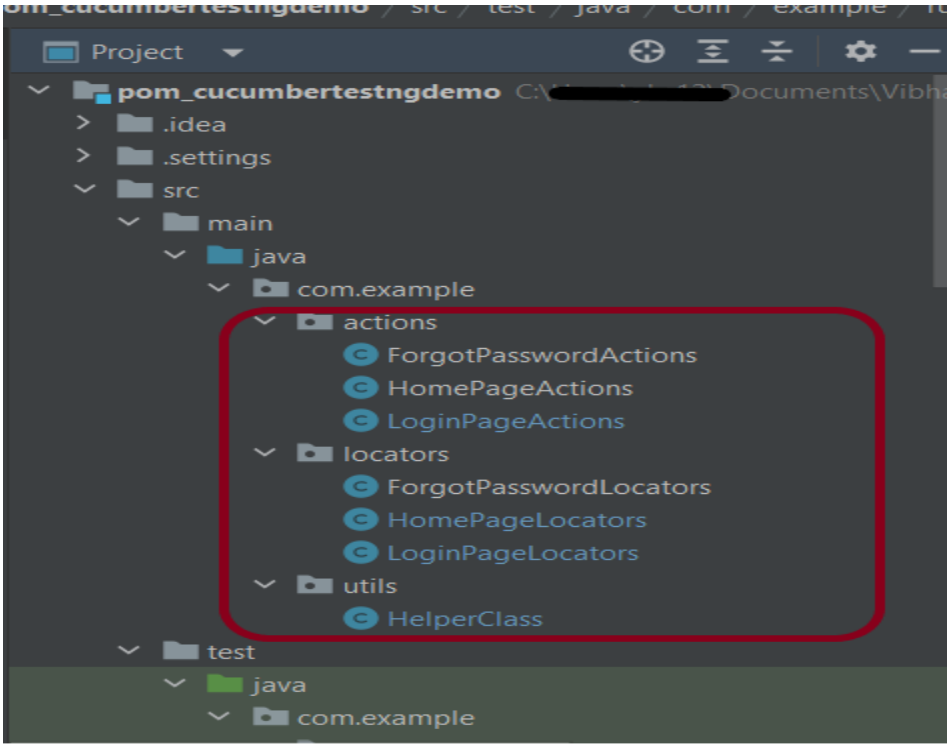
Step 8 – Create a feature file in the src/test/resources

Create a folder with name features. Now, create the feature file in this folder. The feature file should be saved with the extension .feature. This feature file contains the test scenarios created to test the application. The Test Scenarios are written in Gherkins language in the format of Given, When, Then, And, But.




Step 9 – Create the classes for locators, actions, and utilities in src/main/java

Create folders – actions, locators, and utils in src/main/java.




Create a Java Class for each page where define WebElements as variables using Annotation @FindBy. Create another Java class that contains methods for actions performed on WebElements. Here, I'm going to create 2 classes for locators – LoginPageLocators and HomePageLocators as well as 2 classes for actions – LoginPageActions and HomePageActions

LoginPageLocators:

```
import org.openqa.selenium.WebElement;

import org.openqa.selenium.support.FindBy;

public class LoginPageLocators {

    @FindBy(name = "username")

    public WebElement userName;

    @FindBy(name = "password")

    public WebElement password;

    @FindBy(xpath = "//*[@id='app']/div[1]/div/div[1]/div/div[2]/div[2]/form/div[1]/div/span")

    public WebElement missingUsernameErrorMessage;

    @FindBy(xpath = "//*[@id='app']/div[1]/div/div[1]/div/div[2]/div[2]/form/div[3]/button")
```

```
    public WebElement login;
    @FindBy(xpath = "//*[@id='app']/div[1]/div/div[1]/div/div[2]/div[2]/div/div[1]/div[1]/p")
    public  WebElement errorMessage;
}
```

HomePageLocators.

```
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
public class HomePageLocators {
      @FindBy(xpath = "//span[@class='oxd-topbar-header-breadcrumb']/h6")
      public  WebElement homePageUserName;
```

Create the action classes for each web page. These action classes contain all the methods needed by the step definitions. In this case, I have created 2 action classes –
LoginPageActions, HomePageActions

LoginPageActions

```
import org.example.locators.LoginPageLocators;
import org.example.utils.HelperClass;
import org.openqa.selenium.support.PageFactory;
public class LoginPageActions {
    LoginPageLocators loginPageLocators = null;
    public LoginPageActions() {
        this.loginPageLocators = new LoginPageLocators();
        PageFactory.initElements(HelperClass.getDriver(),loginPageLocators);
    }
    // Get the error message when username is blank
    public String getMissingUsernameText() {
        return loginPageLocators.missingUsernameErrorMessage.getText();
    }
    // Get the Error Message
    public String getErrorMessage() {
```

```
        return loginPageLocators.errorMessage.getText();
    }
    public void login(String strUserName, String strPassword) {
        // Fill user name
        loginPageLocators.userName.sendKeys(strUserName);
        // Fill password
        loginPageLocators.password.sendKeys(strPassword);
        // Click Login button
        loginPageLocators.login.click();
    }
```

HomePageActions

```
import org.example.locators.HomePageLocators;
import org.example.utils.HelperClass;
import org.openqa.selenium.support.PageFactory;
public class HomePageActions {
    HomePageLocators homePageLocators = null;
    public HomePageActions() {
        this.homePageLocators = new HomePageLocators();
        PageFactory.initElements(HelperClass.getDriver(),homePageLocators);
    }
    public String getHomePageText() {
        return homePageLocators.homePageUserName.getText();
    }
}
```

Create a Helper class where we are initializing the web driver, initializing the web driver wait, defining the timeouts, and creating a private constructor of the class, it will declare the web driver, so whenever we create an object of this class, a new web browser is invoked.

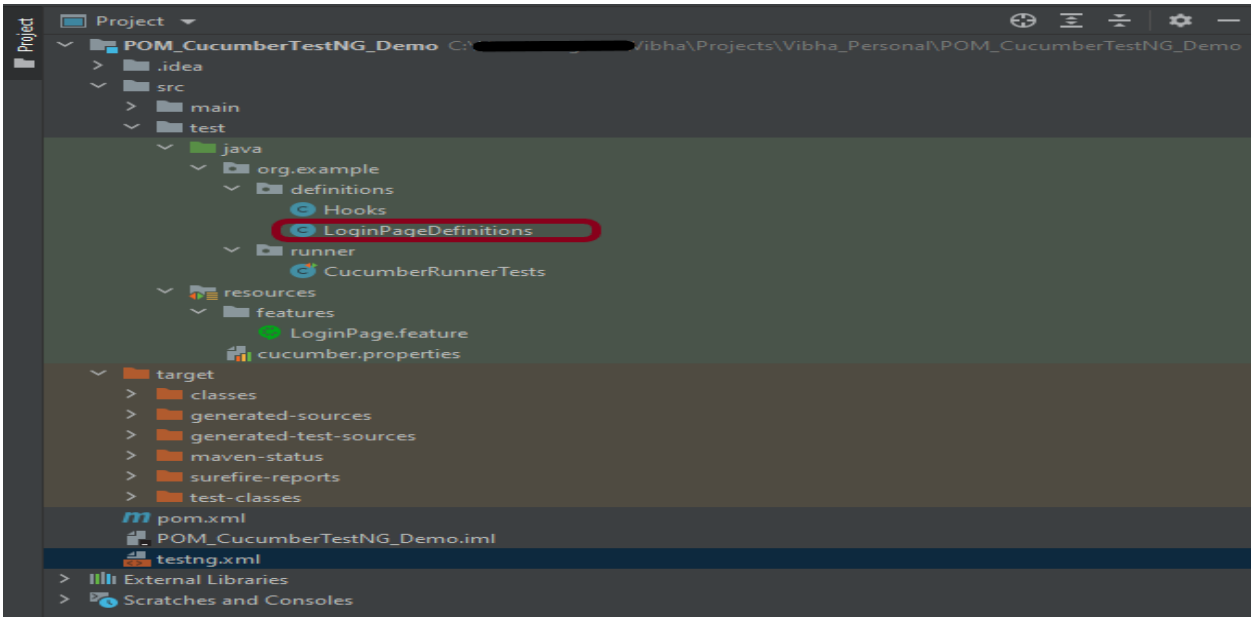
```
package com.example.utils;
import java.time.Duration;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.chrome.ChromeOptions;
public class HelperClass {
    private static HelperClass helperClass;
       private static WebDriver driver;
       public final static int TIMEOUT = 5;
        private HelperClass() {
          WebDriverManager.chromedriver().setup();
          ChromeOptions options = new ChromeOptions();
          options.addArguments("--start-maximized");
          driver = new ChromeDriver(options);
          driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(TIMEOUT));
        }
       public static void openPage(String url) {
          driver.get(url);
       }
       public static WebDriver getDriver() {
          return driver;}
public static void setUpDriver() {
          if (helperClass==null) {
             helperClass = new HelperClass();
          }
       }
       public static void tearDown() {
          if(driver!=null) {
              driver.close();
```

```
            driver.quit();
        }
        helperClass = null;
    }
}
```

Step 10 – Create a StepDefinition class in src/test/java

Create a Java Class called Definition where we will create the Test Code related to the Given, When, Then of Feature file in src/test/java.



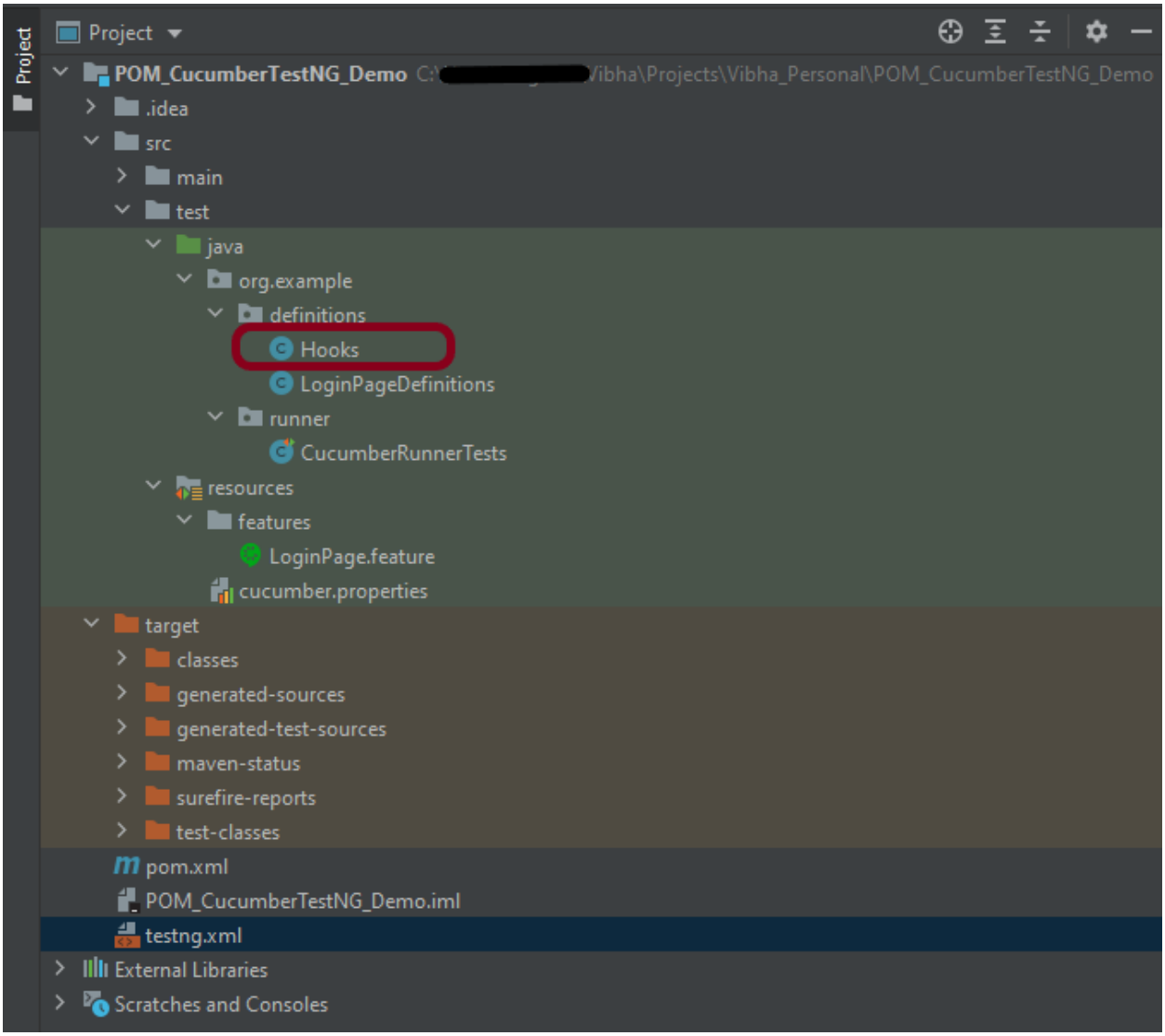

Now, we need to create the Step Definition of the Feature File – LoginPageDefinitions.java.

```
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.example.actions.HomePageActions;
import org.example.actions.LoginPageActions;
import org.example.utils.HelperClass;
import org.testng.Assert;
public class LoginPageDefinitions {
    LoginPageActions objLogin = new LoginPageActions();
```

```
HomePageActions objHomePage = new HomePageActions();
@Given("User is on HRMLogin page {string}")
public void loginTest(String url) {
   HelperClass.openPage(url);
}
@When("User enters username as {string} and password as {string}")
public void goToHomePage(String userName, String passWord) {
   objLogin.login(userName, passWord);
}
@Then("User should be able to login successfully and new page open")
public void verifyLogin() {
   // Verify home page
   Assert.assertTrue(objHomePage.getHomePageText().contains("Dashboard"));
}
@Then("User should be able to see error message {string}")
public void verifyErrorMessage(String expectedErrorMessage) {\
   // Verify error message
   Assert.assertEquals(objLogin.getErrorMessage(),expectedErrorMessage);
}
@Then("User should be able to see a message {string} below Username")
public void verifyMissingUsernameMessage(String message) {
   Assert.assertEquals(objLogin.getMissingUsernameText(),message);
}}
```

Step 11 – Create a Hook class in src/test/java

Create the hook class that contains the Before and After hook to initialize the web browser and close the web browser. I have added the code to take the screenshot of the failed scenario in @After Hook.



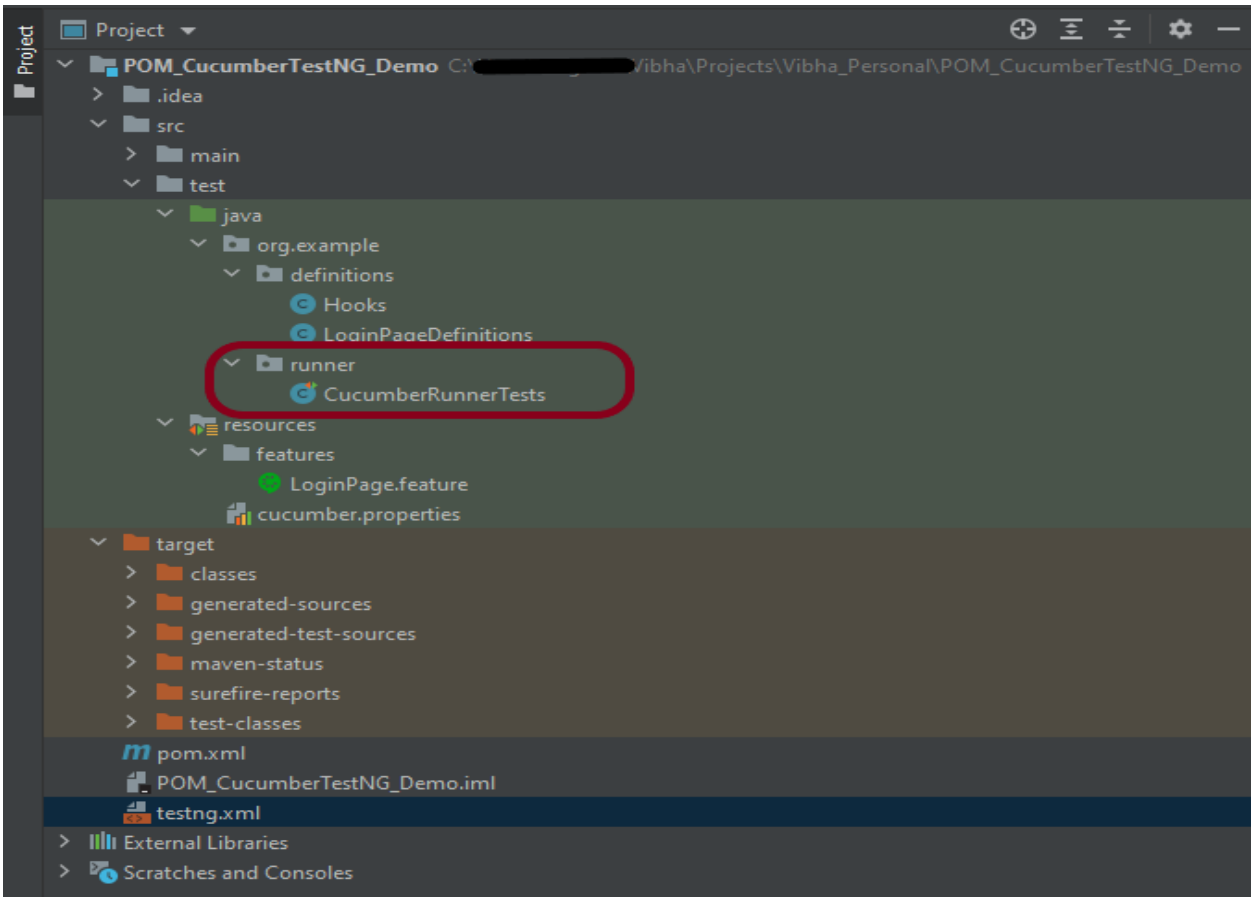

Hooks

```
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import com.example.utils.HelperClass;
import io.cucumber.java.After;
import io.cucumber.java.Before;
import io.cucumber.java.Scenario;
public class Hooks {
    @Before
    public static void setUp() {
```

```
    HelperClass.setUpDriver(); }
  @After
  public static void tearDown(Scenario scenario) {
    //validate if scenario has failed
    if(scenario.isFailed()) {
      final byte[] screenshot = ((TakesScreenshot)
HelperClass.getDriver()).getScreenshotAs(OutputType.BYTES);
      scenario.attach(screenshot, "image/png", scenario.getName()); }
   HelperClass.tearDown();}}
```

Step 12 – Create a TestNG Cucumber Runner class in the src/test/java

Cucumber needs a TestRunner class to run the feature files. It is suggested to create a folder with the name of the runner in the src/test/java directory and create the Cucumber TestRunner class in this folder. Below is the code of the Cucumber TestRunner class.



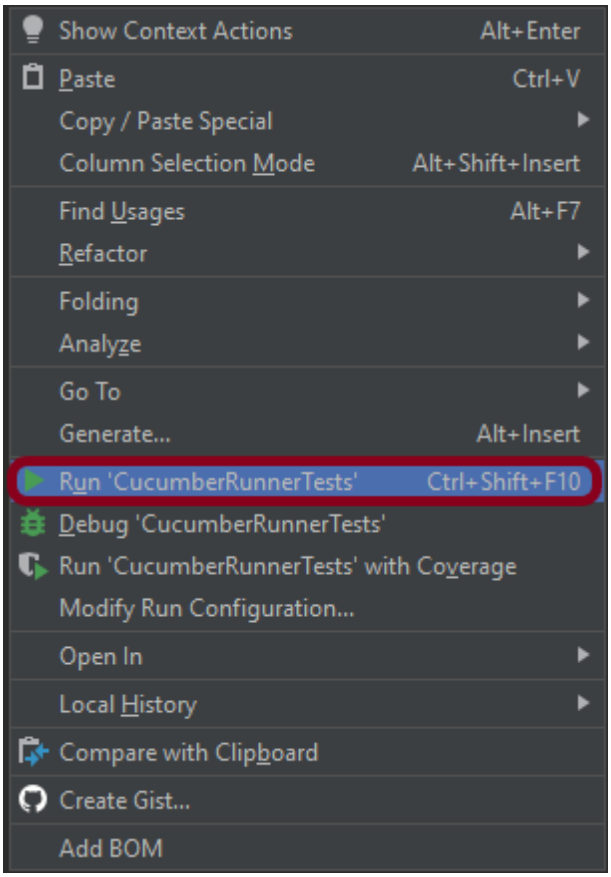
CucumberRunnerTests

```
import io.cucumber.testng.AbstractTestNGCucumberTests;

import io.cucumber.testng.CucumberOptions;

@CucumberOptions(tags = "", features = "src/test/resources/features/LoginPage.feature", glue =
"org.example.definitions",

        plugin = {})

public class CucumberRunnerTests extends AbstractTestNGCucumberTests {

}
```

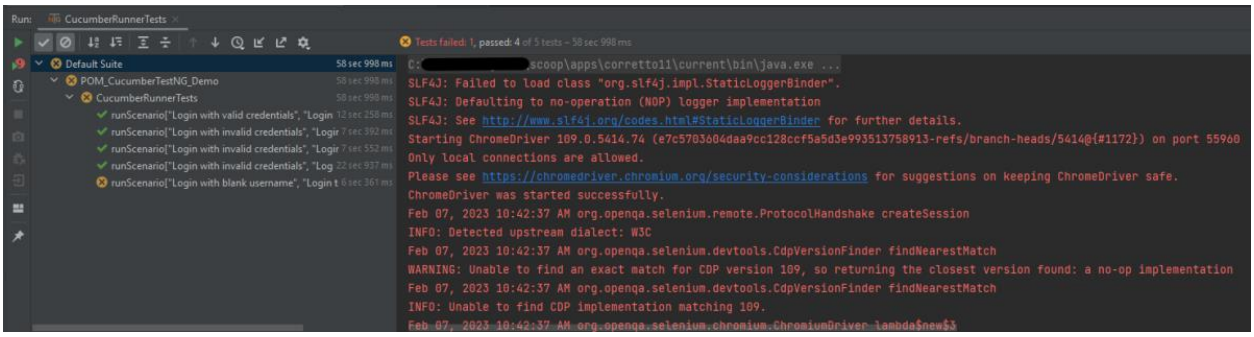
Step 13 – Run the tests from TestNG

You can execute the test script by right-clicking on TestRunner class -> Run As TestNG. (Eclipse)

In the case of the IntelliJ project, right-click on the runner class and select Run 'CucumberRunnerTests'.




The output of the above program is

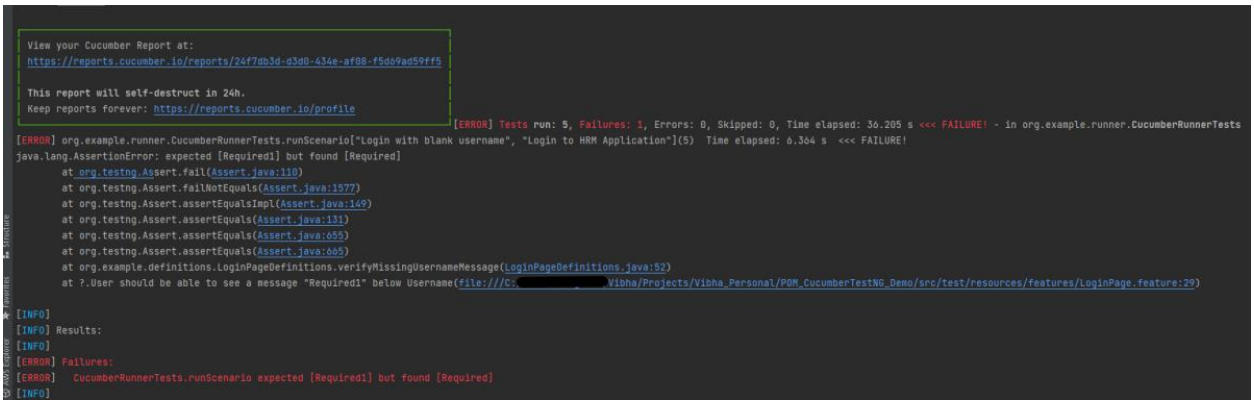



Step 15 – Run the tests from Command Line

Run the below command in the command prompt to run the tests and to get the test execution report.

1mvn clean test

The output of the above program is

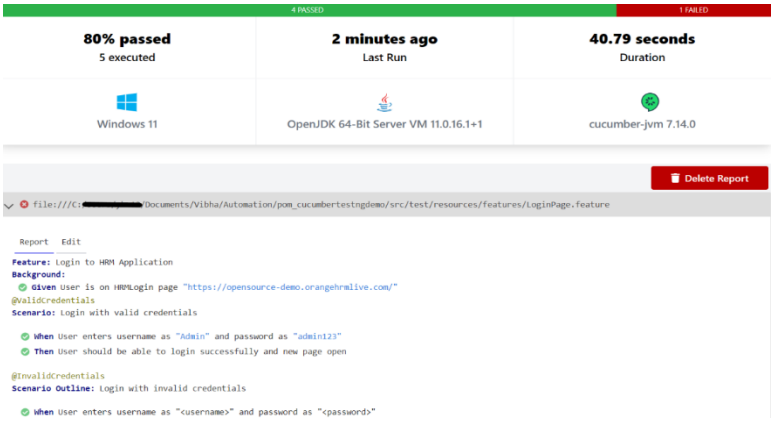



Step 16 – Cucumber Report Generation

To get Cucumber Test Reports, add cucumber.properties under src/test/resources and add the below instruction in the file.

1cucumber.publish.enabled=true

Below is the image of the Cucumber Report generated using the Cucumber Service.




In the above example, as we can see, one of the tests has failed. So, when a test fails, we have written the code to take a screenshot of the failed step. The Attached Image shows the image of the failed test. You can click on that to see the screenshot.

@InvalidCredentials
**Scenario Outline:** Login with invalid credentials

- When User enters username as "<username>" and password as "<password>"
- Then User should be able to see error message "<errorMessage>"

**Examples:**

| username | password | errorMessage |
|---|---|---|
| Admin | admin12$$ | Invalid credentials |
| admin$$ | admin123 | Invalid credentials |
| abc123 | xyz$$ | Invalid credentials |

@MissingUsername
**Scenario Outline:** Login with blank username

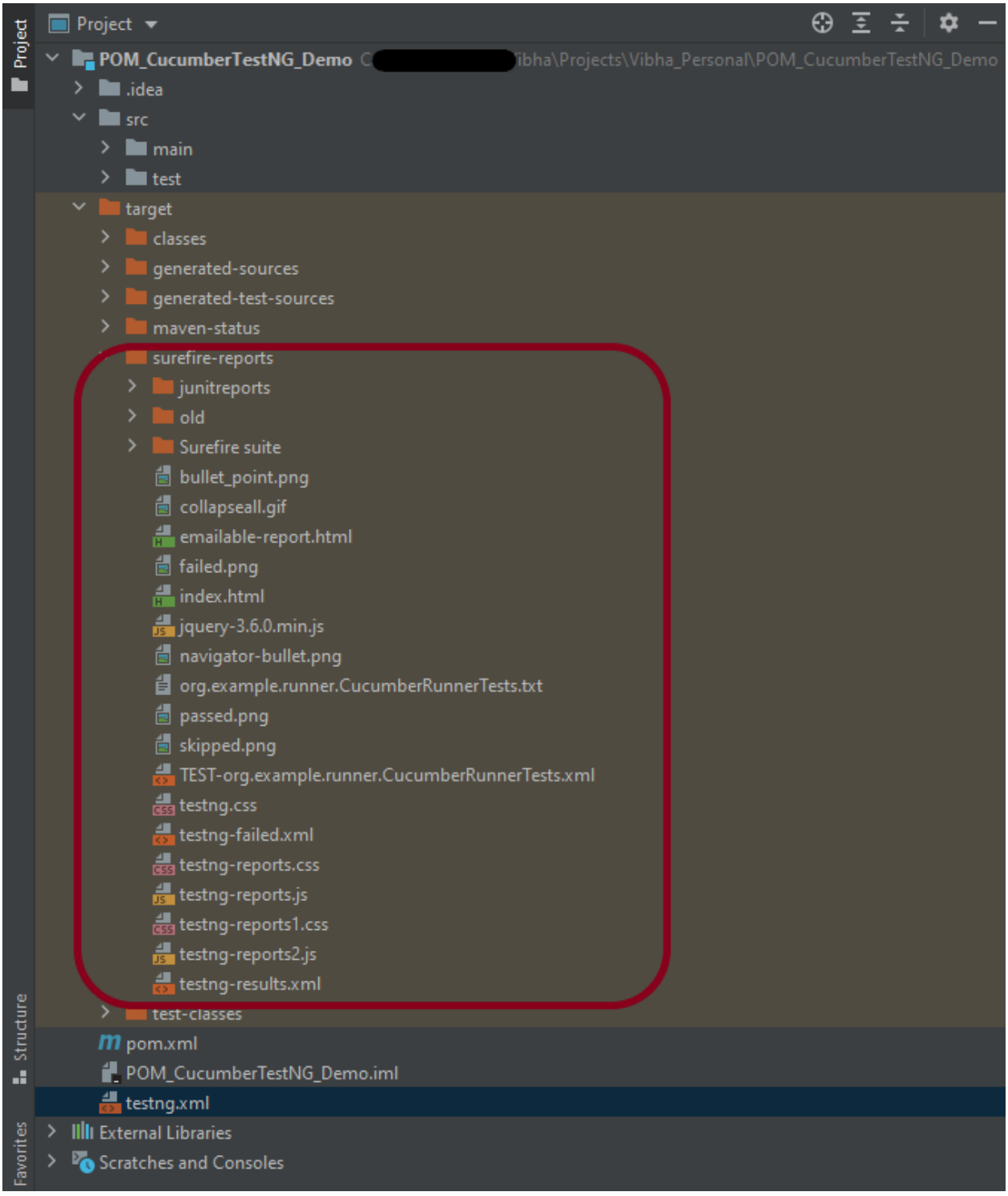
- When User enters username as " " and password as "admin123"
- Then User should be able to see a message "Required1" below Username

```
java.lang.AssertionError: expected [Required1] but found [Required]
    at org.testng.Assert.fail(Assert.java:110)
    at org.testng.Assert.failNotEquals(Assert.java:1577)
    at org.testng.Assert.assertEqualsImpl(Assert.java:149)
    at org.testng.Assert.assertEquals(Assert.java:131)
    at org.testng.Assert.assertEquals(Assert.java:655)
    at org.testng.Assert.assertEquals(Assert.java:665)
    at org.example.definitions.LoginPageDefinitions.verifyMissingUsernameMessage(LoginPageDefinitions.java:52)
    at ?.User should be able to see a message "Required1" below
Username(file:///C:/Users/SingVi04/Vibha/Projects/Vibha_Personal/POM_CucumberTestNG_Demo/src/test/resources/features/LoginPage.feature:29)
```

▼ Attached Image


Step 17 – TestNG Report Generation

TestNG generates various types of reports under the target->surefire-reports folder like emailable-report.html, index.html, testng-results.xml.




We are interested in the “emailable-report.html” report. Open “emailable-report.html“, as this is an HTML report, and open it with the browser. The below image shows emailable-report.html.

emailable-report.html

| Test | # Passed | # Skipped | # Retried | # Failed | Time (ms) | Included Groups | Excluded Groups |
|---|---|---|---|---|---|---|---|
| Suite | | | | | | | |
| Cucumber6 with TestNG | 4 | 0 | 0 | 1 | 49,301 | | |

| Class | Method | Start | Time (ms) |
|---|---|---|---|
| Suite | | | |
| Cucumber6 with TestNG — failed | | | |
| com.example.runner.CucumberRunnerTests | runScenario | 1697485076843 | 6933 |
| Cucumber6 with TestNG — passed | | | |
| com.example.runner.CucumberRunnerTests | runScenario | 1697485050146 | 9197 |
| | runScenario | | |
| | runScenario | | |
| | runScenario | | |

**Cucumber6 with TestNG**

**:om.example.runner.CucumberRunnerTests#runScenario**

| Parameter #1 | Parameter #2 |
|---|---|
| "Login with blank username" | "Login to HRM Application" |
| **Exception** | |

```
java.lang.AssertionError: expected [Required1] but found [Required]
        at com.example.definitions.LoginPageDefinitions.verifyMissingUsernameMessage(LoginPageDefinitions.java:63)
        at ✱.User should be able to see a message "Required1" below Username(file:///C:[redacted]/Documents/Vibha/Automation/pom_cucumbertestngdemo/src/test/resources/features/LoginPage.feature:29)
... Removed 6 stack frames
```

back to summary

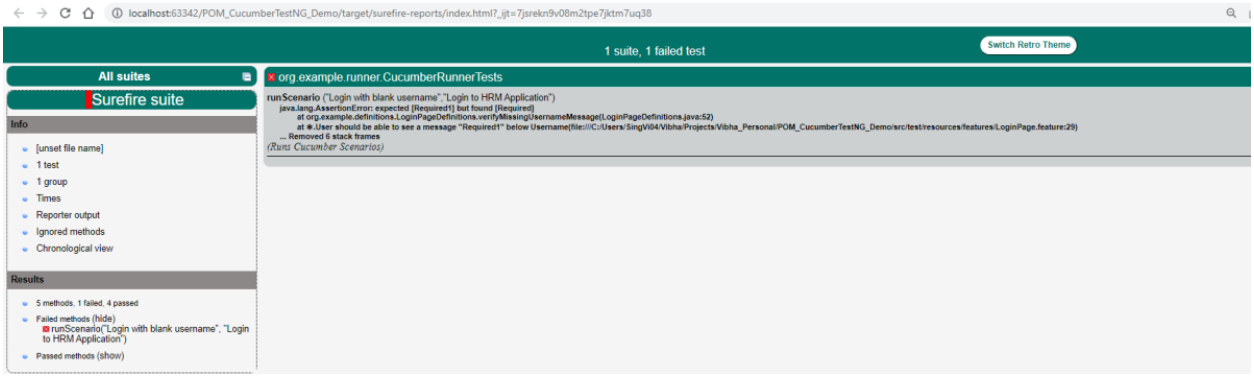
**:om.example.runner.CucumberRunnerTests#runScenario**

| Parameter #1 | Parameter #2 |
|---|---|
| "Login with invalid credentials" | "Login to HRM Application" |

back to summary

Index.html

TestNG also produces an “index.html” report. The below image shows the index.html report.




**Result :**

Thus the Page Object Model has been built using Selenium and TestNG.

**Ex No: 5 c Build BDD framework with Selenium, TestNG and Cucumber**

**Date:**

**Scope:**

To build BDD framework with Selenium, TestNG and Cucumber

**Procedure:**

Step 1:

Set up the dependencies and gradle file configuration

Step 2:

To execute the TestNG suite , include the function

```
@Test(groups = "cucumber", description = "Runs Cucumber Scenarios", dataProvider =
"scenarios")
public void runScenario(PickleWrapper pickleWrapper, FeatureWrapper featureWrapper) {
testNGCucumberRunner.runScenario(pickleWrapper.getPickle());
}
@DataProvider
public Object[][] scenarios() {
return testNGCucumberRunner.provideScenarios();
}
```

Step 3:

Dowload the selenium and initialize the web drivers.

Step 4:

Tests the reset button functionality of guru99 website

```
Feature: Reset Functionality Scenarios

Scenario Outline: Reset Functionality
Given User navigates to "<website>"
When User enter the Username "<username>" and Password "<password>"
Then Reset the credential

Examples:
| website | username | password |
| https://www.demo.guru99.com/v4 | test1 | test1 |
| https://www.demo.guru99.com/v4 | test2 | test2 |
```

Step 4:

Test to check the Home button functionality of javatpoint website

```
Feature: Home Button Functionality Scenarios

Scenario Outline: Home Button Functionality
Given User navigates to "<website>"
When User clicks on Home Button
Then Home website "<website2>" is reached

Examples:
| website | website2 |
| https://www.javatpoint.com/gui-testing | https://www.javatpoint.com/ |
| https://www.javatpoint.com/gui-testing | https://www.javatpoint.com/ |
```

Let's see the implementation now for test1

@Given

```
("^User navigates to \"(.*)\"$")
 public void user_navigates_to_website(String website) throws Exception {
  log.info("User navigates to website: "+ website);
  webdriver.get(website);
  webdriver.manage().window().maximize();
  Thread.sleep(2000);
 }
```

@When

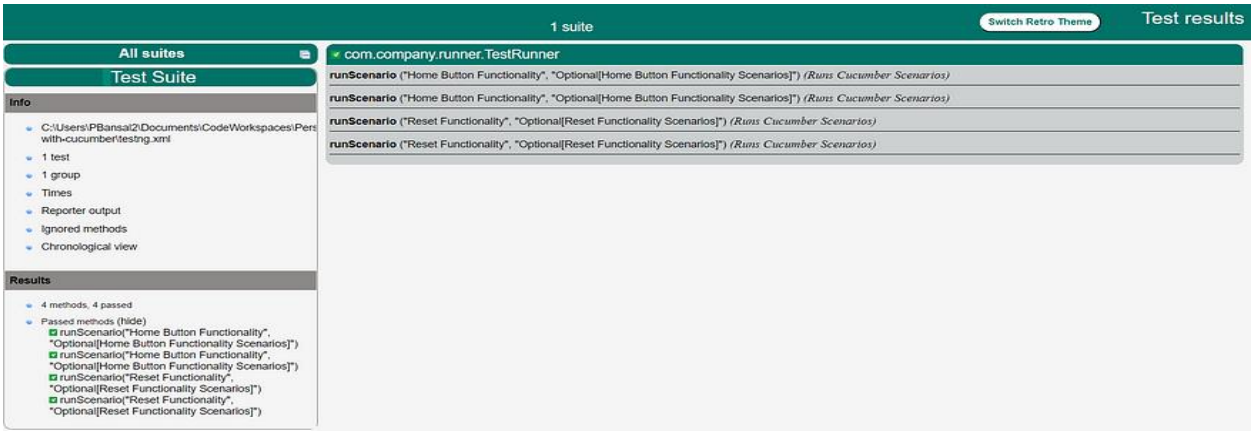
```
("^User enter the Username \"(.*)\" and Password \"(.*)\"$")
 public void enter_the_Username_and_Password(String username,String password) throws
Throwable
 {
  webdriver.findElement(By.xpath("//input[@name='uid']")).sendKeys(username);
  webdriver.findElement(By.xpath("//input[@name ='password']")).sendKeys(password);
 }@Then ("^Reset the credential$")
 public void reset_the_credential() throws InterruptedException {
  webdriver.findElement(By.xpath("//input[@name ='btnReset']")).click();
 }
```

For test2, here is the implementation

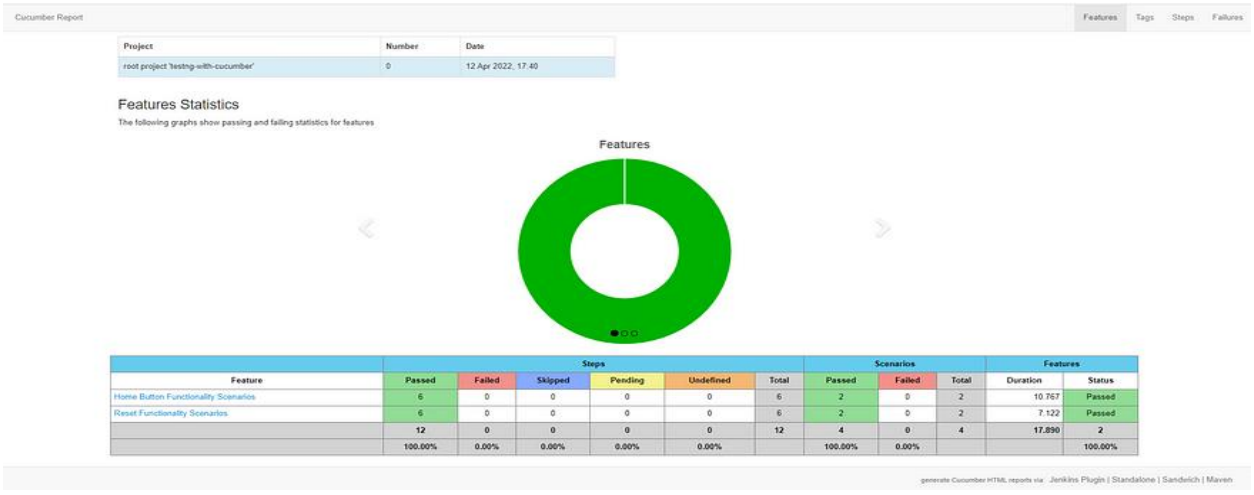
```
@When("^User clicks on Home Button$")
public void user_clicks_on_home_button() throws Throwable
{
webdriver.findElement(By.xpath("//a[normalize-space()='Home']")).click();
}

@Then("^Home website \"(.*)\" is reached$")
public void home_website_is_reached(String expectedUrl) throws InterruptedException {
String url = webdriver.getCurrentUrl();
Assert.assertEquals(expectedUrl, url);
}
```

Output:

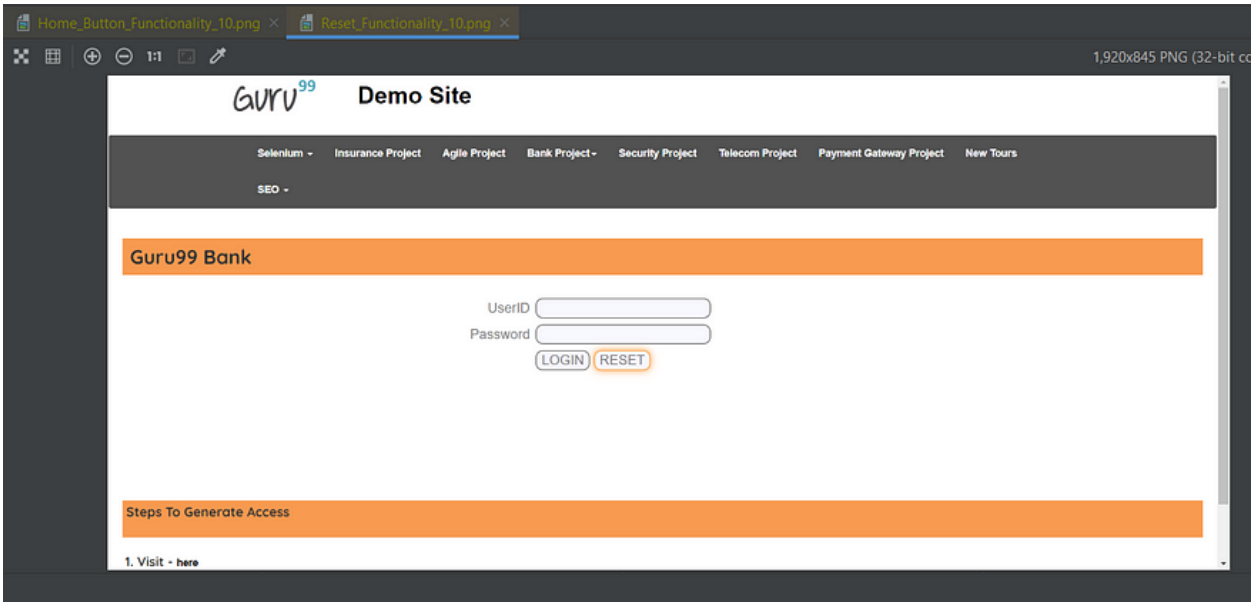
TestNG reports looks like this




Cucumber reports looks like this




And screenshots also got captured




**Result:**

Thus the behaviour-driven development framework has been build using Selenium, TestNG, Cucumber.