**Cocumber Assisted Practice**

**Question 1:** You are given a project to explain the basics of Cucumber setup (Maven, Extent Reports, and Cucumber Eclipse Plugin).

This section will guide you to:

* Set up Cucumber eclipse plugin
* Set up Maven project
* Set up Extent report

**Development Environment:**

* JRE: OpenJDK Runtime Environment 11.0.2
* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* TestNG
* Selenium jars
* Cucumber jars

This guide has four subsections, namely:

5.1.1 Configuring Cucumber Eclipse

5.1.2 Configuring Cucumber with Maven project

5.1.3 Configuring Cucumber extent report

5.1.4 Pushing the code to your GitHub repositories

**Step 5.1.1:** Configuring Cucumber Eclipse

Cucumber jar files are already present in your practice lab. Refer to the lab guide for Phase 2 for more information.

* Launch Eclipse.
* Create Java Project.
* Go to Java Project and create a folder “jars.”
* Add all the downloaded jars to the “Jars” folder.
* Select the Created Project → Right Click → Build path → Configure Build path →Click on Libraries → Add JARs →Select all the jars from “Jars” folder → Click on Apply → Add Library → Select TestNG → Click on Apply → Click on Finish.

**Step 5.1.2:** Configuring Cucumber with MavenProject

* Create a Maven Project.
* Go to File → New → Others → Maven → Maven Project → Next.
* Provide group ID (group ID will identify your project uniquely across all projects).
* Provide artifact ID (artifact ID is the name of the jar without version. You can choose any name, which is in lowercase). Click on Finish.
* Open pom.xml.
* Go to the package explorer on the left-hand side of Eclipse.
* Expand the created project.
* Locate the pom.xml file.
* Right-click and select the option, open with “Text Editor.”
* Adding dependencies to the project: This will indicate Maven about the jar files to be downloaded from the central repository to the local repository.
* Open the pom.xml in the edit mode, create dependencies tag (<dependencies></dependencies>), inside the project tag.
* Inside the dependencies tag, create dependency tag (<dependency></dependency>).
* Copy the dependency tag for the following from Maven Repository.
* Selenium Webdriver
* Cucumber-Core jar
* Cucumber-Java jar
* Cucumber-TestNG jar
* Provide the information of all copied dependencies within the dependency tag.
* Verify binaries.
* Once the pom.xml is edited successfully, save it.
* Go to Project → Clean. After a few minutes, you will be able to see a Maven repository.

**Step 5.1.3:** ConfiguringCucumber Extent Report

* Add below Cucumber Extent Report library to Maven project.

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.0.6</version>

</dependency>

<dependency>

<groupId>com.vimalselvam</groupId>

<artifactId>cucumber-extentsreport</artifactId>

<version>3.0.2</version>

</dependency>

* Add Extent Config to the project.

Extent Config is required by the Cucumber Extent Report plugin to read the report configuration. This gives the capability to set many useful settings to the report from the *XML* configuration file.

* Create a ***New File*** and name it as ***extent-config.xml*** by right clicking on the ***configs*** folder in the project.
* Read the extent-config.xml path.
* Make an entry for the path of the config in the ***Configuration.properties*** file.

*reportConfigPath=C:/ToolsQA/CucumberFramework/configs/extent-config.xml*

* Write a method ***getReportConfigPath()*** in the ***ConfigFileReader***class to return the extent report config file path.

public String getReportConfigPath(){

String reportConfigPath = properties.getProperty("reportConfigPath");

if(reportConfigPath!= null) return reportConfigPath;

else throw new RuntimeException("Report Config Path not specified in the Configuration.properties file for the Key:reportConfigPath");

}

* Modify TestRunner to Implement Cucumber Extent Reporter
* Modify the runner class and add *com.cucumber.listener.ExtentCucumberFormatter:output/report.html* as a plugin followed by the report file as input. This should be done within the *@CucumberOptions* annotation.

***@CucumberOptions( plugin = {“com.cucumber.listener.ExtentCucumberFormatter:target/cucumber- reports/report.html”})***

The above setup will generate the report in the output directory with the name report.html.

* Write extent reports

Add a method *writeExtentReport()* in the *TestRunner* class to write the report.

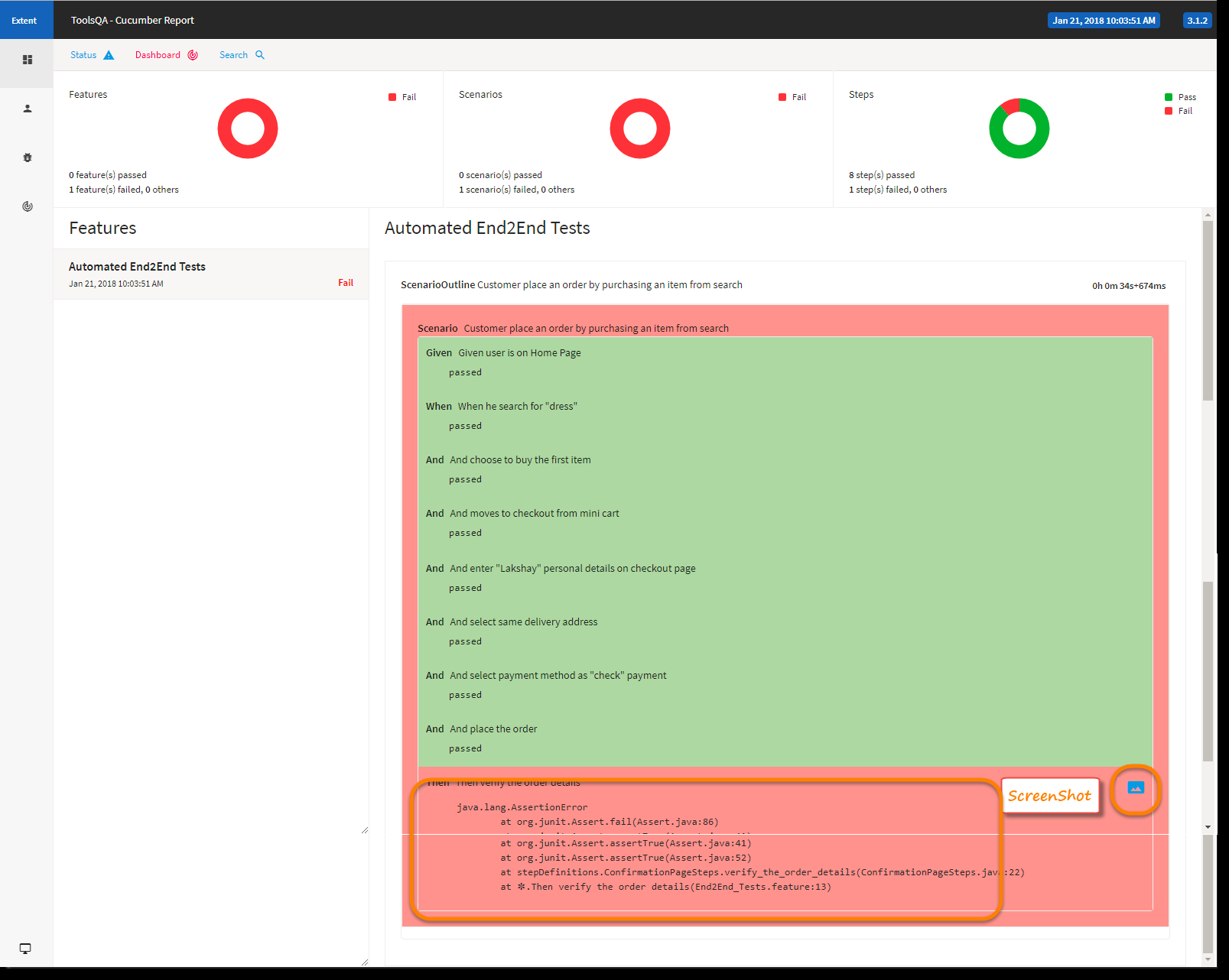
@AfterClass

public static void writeExtentReport() {

Reporter.loadXMLConfig(new File(FileReaderManager.getInstance().getConfigReader().getReportConfigPath()));

}

* Sample extent report



Question 2: You are given a project to define the functions of various Gherkin keywords.

5.2Gherkin Keywords

This section will guide you to :

* Understand the types of Gherkin keywords

**Development Environment:**

* JRE: OpenJDK Runtime Environment 11.0.2
* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* TestNG
* Selenium jars
* Cucumber jars

This guide has two subsections, namely:

5.2.1 Demonstrating Gherkin keywords with example

**Step 5.2.1:** Demonstrating Gherkin keywords with example.

The following are the keywords used in Gherkin:

* **Feature**: This gives information on the high-level business functionality and the purpose of application under test. Everybody should be able to understand the intent of feature file by reading the first feature step.

Feature: LogIn Action Test

Description: This feature will test a LogIn and LogOut functionality

* **Scenario**: Basically, a scenario represents a particular functionality which is under test. By seeing the scenario, user should be able to understand the intent behind the scenario and test.

Scenario: Successful Login with Valid Credentials

Given User is on Home Page

When User Navigate to LogIn Page

And User enters UserName and Password

Code: Feature: Testing Ninjademo login page

Scenario: User should open NinjaDemo login page on ChromeBrowser and get title of the page

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

Then User gets title of the page

And user closes the browser

Question 3

You are given a project to demonstrate Gherkin: Given, When, Then, and Background Steps

@all

Feature: Testing Ninjademo login page

@homepage @sanity

Scenario: User should open NinjaDemo login page on ChromeBrowser and get title of the page

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

Then User gets title of the page

And user closes the browser

@search @regression

Scenario: User should search for a product in the hoempage

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And User will enter a product in search box

And user click on submit button

Then product list should be displayed

And user closes the browser

@cart

Scenario: User Click on Add to Cart link

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And user click on Addto cart link

Then user is navigated to Cart page

And user closes the browser

Question 4:- You are given a project to explain Step Arguments in Gherkin.

**:** Gherkin step argument

* In some cases, you might want to pass more data to a step that fits on a single line. For this purpose, Gherkin has Docstrings and Data tables.

**Step 5.4.2:** Docstrings

* If you need to specify information in a scenario that won't fit on a single line, you can use Docstrings.
* A Docstring follows a step. It starts and ends with three double quotes, like this:

|  |
| --- |
| When I ask to reset my password  Then I should receive an email with:  """  Dear bozo,  Please click this link to reset your password  """ |

**Step 5.4.3:** Data Tables

* Data Tables are handy for passing a list of values to a step definition

|  |
| --- |
| | name |email | twitter |  | Aslak | aslak@cucumber.io | @aslak\_hellesoy |  | Julien | julien@cucumber.io | @jbpros |  | Matt | matt@cucumber.io | @mattwynne | |

* Just like Docstrings, Data Tables will be passed to the step definition as the last argument.

Question 5:- You are given a project to demonstrate the execution of a step definition file.

**Step 5.5.1:** Gherkin comments with example.

* Comment is basically a piece of code meant for documentation purposes and not for execution.
* Feature file: In case of a feature file, we just need to put # before beginning your comment.

**Example:**

|  |
| --- |
| Feature: annotation  #This is how background can be used to eliminate duplicate steps  Background:  User navigates to Facebook  Given I am on Facebook login page |

* Step definition file: If you are using Java as a platform then mark your comments with //.

**Example:**

|  |
| --- |
| //scroll to the bottom of the page  ((JavascriptExecutor) driver).executeScript("window.scrollTo(0, document.body.scrollHeight)"); |

**Step 5.5.2:** Gherkin tags with example.

* if we have many scenarios in a feature file, to put them under a single umbrella, we use tags to generate reports for specific scenarios under the same tag.
* Tags are defined in our runner class like this:

|  |
| --- |
| @RunWith(Cucumber.class)  @CucumberOptions{  format= {"Pretty" ,"json:target/output.json", "html:targer/html"},  feature={"src/functional-test/resources"},  tags={"@tag", "@tag1"} } |

* When we define multiple tags in runner class in below form, it will be defined with the use of logical operator:
  + tags = {“@tag”, “@tag1”}: means AND condition. It says that scenarios matching both these tags need to be executed.
  + tags = {“@tag1, @tag2”}: means OR condition. It says that scenarios matching any of this tag need to be executed.

Code:- package runner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features=

"C:\\Users\\sadiq\\eclipse-workspace\\cocumber\_practise\\src\\test\\java\\features\\hooksdemo1.feature ",

glue = {"steps"}, // packagename where the steps are

plugin= {"pretty","html:target/cucumberreport.html"},

//dryRun=true

// tags="@sanity", // all the scenario is feature file with tagname sanity will get executed

//tags="@regression and @search" // and operator// scanerios whicha re having both of thes etags will run

//tags="@regression or @sanity"

// tags="@all" // all scnaerios will run

//tags = "not @search" // run all the scenarios except scenario with tag search

tags="@regression"

)

public class hooksdemo1runner {

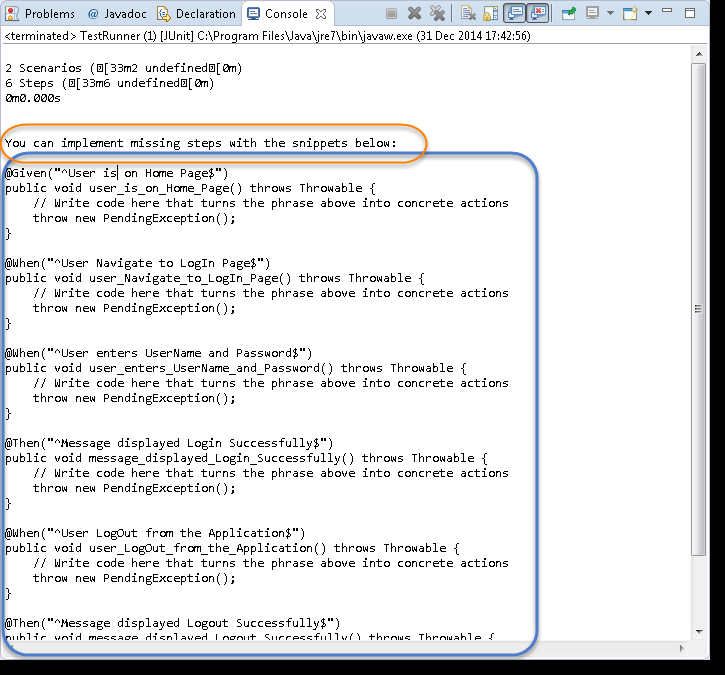
// we dont write anythign over here.

}

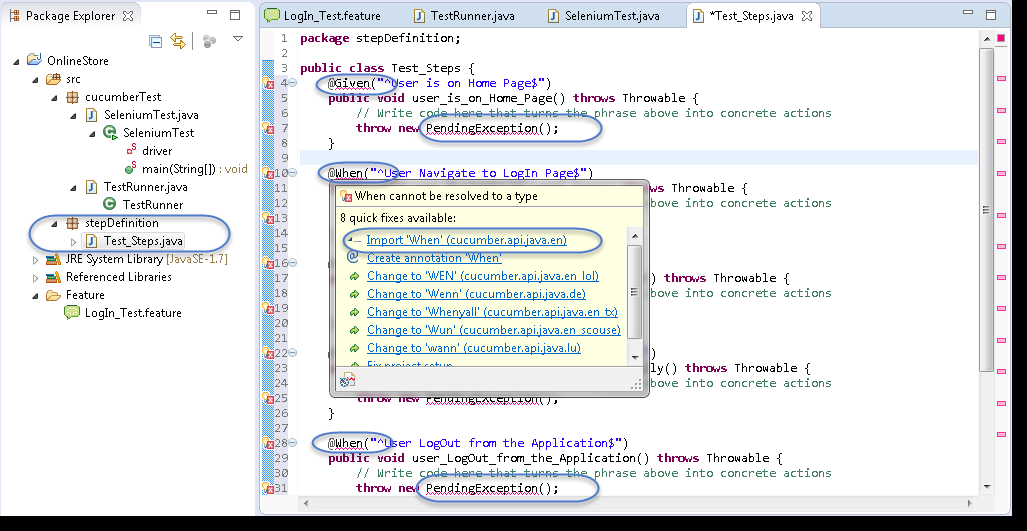
Question 6:- You are given a project to demonstrate the execution of a step definition file.

**1** Adding a step definition file.

* Create a new Class file in the “stepDefinition” package and name it as “Test\_Steps”
* When we run the Feature file as Cucumber feature, the following message will be displayed in the console window:



* Copy the complete text marked in a blue box and paste it into the above created Test\_Steps class.
* The test will show many errors on “@” annotations. At the annotations, import the “cucumber.api.java.en” for all the annotations.



## **Step 5.6.2:** Providing implementations using Selenium Java for the Step Definition methods.

* Using Selenium Java, provide the implementation to the first method “@Given (“^User is on Home Page$”)”
* Launch the Browser
* Navigate to Home Page

Method will look like this now:

|  |
| --- |
| **@Given("^User is on Home Page$")**  **public void user\_is\_on\_Home\_Page() throws Throwable**  **{**  **driver = new FirefoxDriver();**  **driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS); driver.get("http://www.store.demoqa.com");**  **}** |

## 

* Using Selenium Java, provide the implementation to the second method “@When (“^User Navigate to Login Page$”)”
* Click on the Login link

Method will look like this now:

|  |
| --- |
| @When("^User Navigate to Login Page$")  public void user\_Navigate\_to\_Login\_Page() throws Throwable  {  driver.findElement(By.xpath(".//\*[@id='account']/a")).click();  } |

* Using Selenium Java, provide the implementation to the second method.

“@When (“^User enters UserName and Password$”)”

* Enter UserName and Password
* Click on Submit button

Method will look like this now:

|  |
| --- |
| @When("^User enters Username and Password$")  public **void** user\_enters\_Username\_and\_Password() throws Throwable  {  driver.findElement(By.id("log")).sendKeys("testuser\_1");  driver.findElement(By.id("pwd")).sendKeys("Test@123");  driver.findElement(By.id("login")).click();  } |

* Do the same steps for the rest of the methods and the complete Test\_Steps class will look like this:

|  |
| --- |
| **package stepDefinition;**    **import java.util.concurrent.TimeUnit;**  **import org.openqa.selenium.By;**  **import org.openqa.selenium.WebDriver;**  **import org.openqa.selenium.firefox.FirefoxDriver;**  **import cucumber.api.java.en.Given;**  **import cucumber.api.java.en.Then;**  **import cucumber.api.java.en.When;**  **public class Test\_Steps**  **{ public static WebDriver driver;**  **@Given("^User is on Home Page$")**  **public void user\_is\_on\_Home\_Page() throws Throwable{**  **driver = new FirefoxDriver();**  **driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);**  **driver.get("http://www.store.demoqa.com"); }**    **@When("^User Navigate to LogIn Page$")**  **public void user\_Navigate\_to\_LogIn\_Page() throws Throwable {**  **driver.findElement(By.xpath(".//\*[@id='account']/a")).click(); }**    **@When("^User enters UserName and Password$")**  **public void user\_enters\_UserName\_and\_Password() throws Throwable {**  **driver.findElement(By.id("log")).sendKeys("testuser\_1");**  **driver.findElement(By.id("pwd")).sendKeys("Test@123");**  **driver.findElement(By.id("login")).click(); }**    **@Then("^Message displayed Login Successfully$")**  **public void message\_displayed\_Login\_Successfully() throws Throwable{**  **System.out.println("Login Successfully"); }**    **@When("^User LogOut from the Application$")**  **public void user\_LogOut\_from\_the\_Application() throws Throwable{**  **driver.findElement (By.xpath(".//\*[@id='account\_logout']/a")).click(); }**    **@Then("^Message displayed Logout Successfully$")**  **public void message\_displayed\_Logout\_Successfully() throws Throwable {**  **System.out.println("LogOut Successfully"); }}** |

Question 7:- You are given a project to demonstrate how Before and After hooks are used in a scenario

Before and After Hook

* Hooks are blocks of code that run ***before*** or ***after*** each scenario. You can define them anywhere in your project or step definition layers using the methods ***@Before*** and ***@After.***

|  |
| --- |
| **package utilities;**  **import cucumber.api.java.After;**  **import cucumber.api.java.Before;**    **public class Hooks {**  **@Before**  **public void beforeScenario(){**  **System.out.println("This will run before the Scenario");**  **}**  **@After**  **public void afterScenario(){**  **System.out.println("This will run after the Scenario");A**  **}**  **}** |

Code:-

package steps;

import io.cucumber.java.After;

import io.cucumber.java.Before;

public class TestHooksDemo {

// As the name suggest hooks are used to connect a method to a scenario

// Hooks can be used to determin pre-equiste for the test case execution

// hooks come with annotation =>

// @Before : Annotated method will run before a scenario in feature file

//, @After : Annotated method will run after a scenario in feature file

@Before(value="@regression")

public void init()

{

System.out.println("test execution started");

System.out.println("DB connection started");

}

@After(value="@regression")

public void dbclose()

{

System.out.println("test execution completed");

System.out.println("DB connection closed");

}

}

Question:- You are given a project to demonstrate a tagged scenario

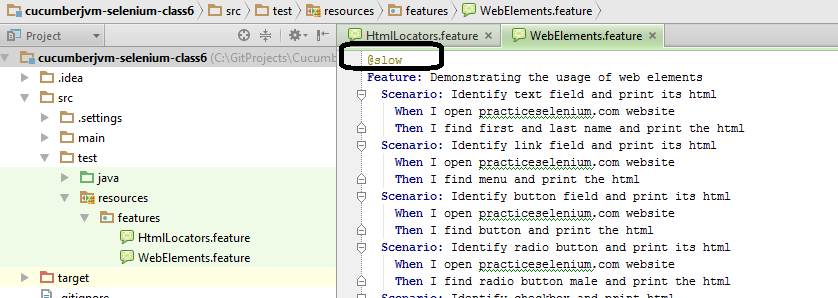
**:** How to add a tag to a feature file and scenario

* Syntax to add tag for feature file

Type “@<tag\_name>” at the top of Feature or Scenario

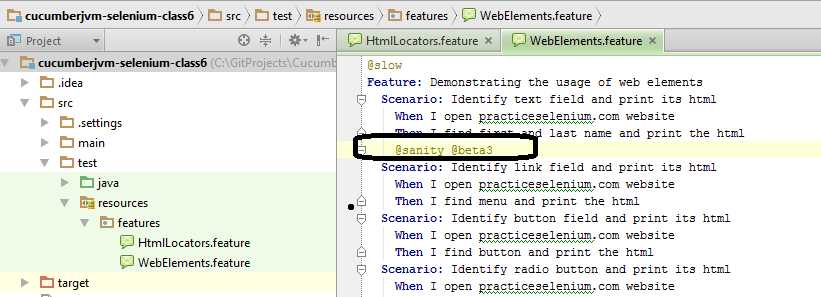
* Tag a feature

The tag applies automatically to all the scenarios on the feature file

****

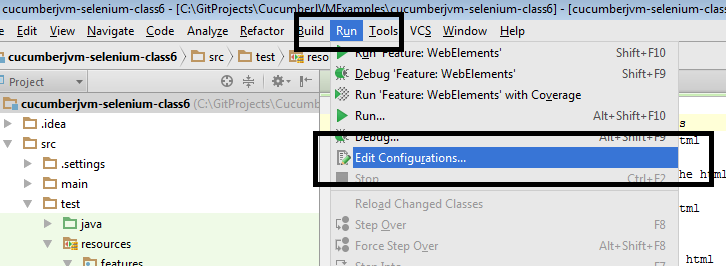
* Tag a scenario

Place the tag at the line exactly above the Scenario.

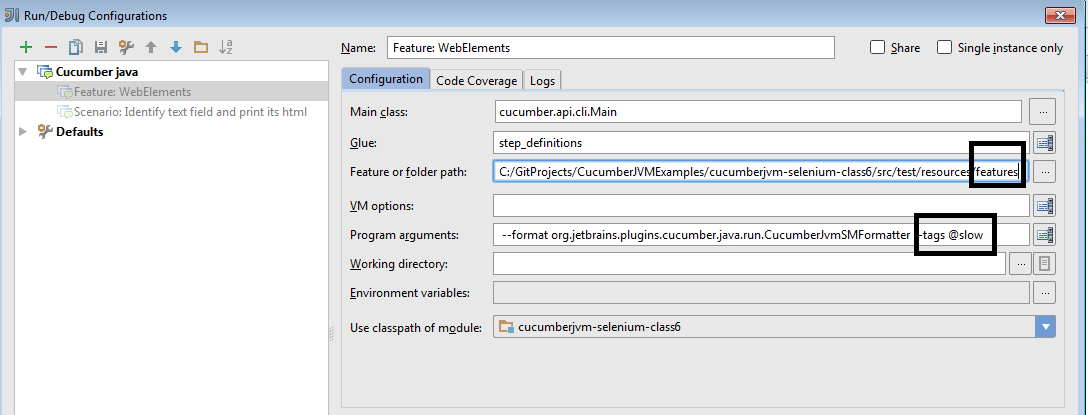


**Step 5.8.2:** Methods to run the tagged file

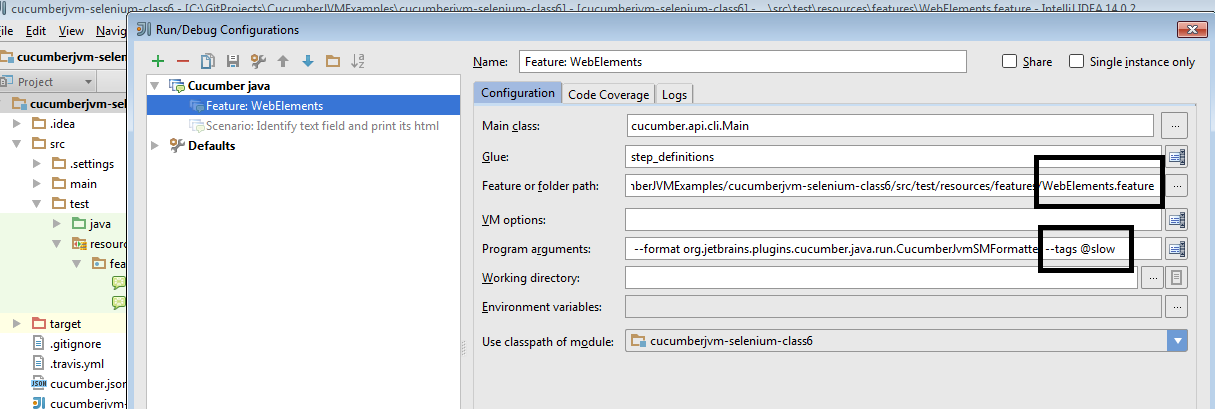
* Method1 (Run all @slow in features directory)



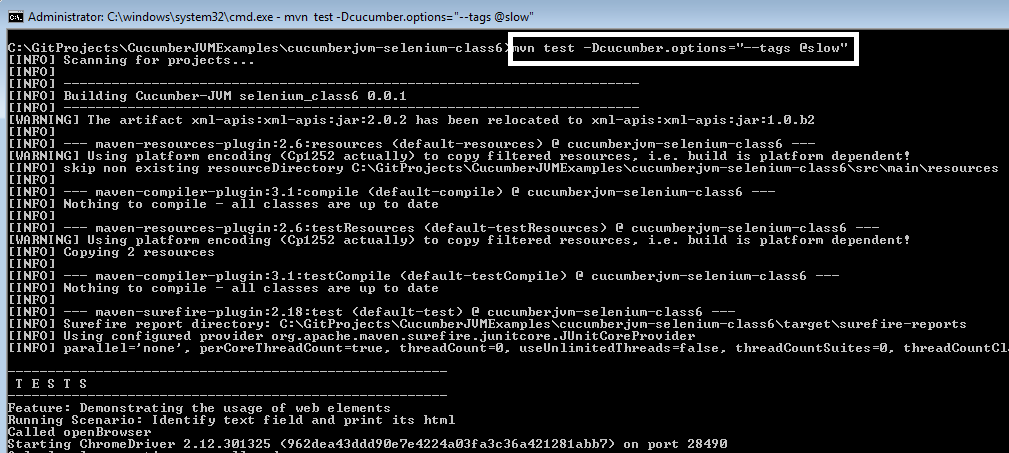
* Paste the path to **feature** folder in your project and write the following in Program arguments:



* Method2 (Run all @slow scenarios in a particular feature)

****

* Method3 (Run from the command line)



Code:- @all

Feature: Testing Ninjademo login page

@homepage @sanity

Scenario: User should open NinjaDemo login page on ChromeBrowser and get title of the page

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

Then User gets title of the page

And user closes the browser

@search @regression

Scenario: User should search for a product in the hoempage

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And User will enter a product in search box

And user click on submit button

Then product list should be displayed

And user closes the browser

@cart

Scenario: User Click on Add to Cart link

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And user click on Addto cart link

Then user is navigated to Cart page

And user closes the browser

Question 9:- You are given a project to demonstrate how to execute multiple scenarios.

Multiple scenarios with examples

* Feature file can have more than one scenario or scenario outline. You can write your all possible requirement or scenarios for a particular feature in a feature file.

|  |
| --- |
| Feature: Registration, Login and MyAccount  Background:  Given I am on the homepage  And I follow "Sign in"  @regression @smoke  Scenario: Verify Login Functionality  And I fill "email address" with "goswami.tarun77@gmail.com"  And I fill "password" with "Test@1234"  And I click "sign in"  Then I should see "MY ACCOUNT" heading  @regression  Scenario: Create New User  When I fill "registration email text box" with "goswami.tarun77+1@gmail.com"  Then I click "create an account button"  And I enter following details  | First Name | Tarun |  | Last Name | Goswami |  | Password | Test1234 |  | Date | 13 |  | Year | 1989 |  And I click "register button" |

Question 10:-

You are given a project to demonstrate how to use tagged hooks

Code:-

@all

Feature: Testing Ninjademo login page

@homepage @sanity

Scenario: User should open NinjaDemo login page on ChromeBrowser and get title of the page

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

Then User gets title of the page

And user closes the browser

@search @regression

Scenario: User should search for a product in the hoempage

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And User will enter a product in search box

And user click on submit button

Then product list should be displayed

And user closes the browser

@cart

Scenario: User Click on Add to Cart link

Given User is on Chrome browser

When User is on Chrome browser users enters the URL

And user click on Addto cart link

Then user is navigated to Cart page

And user closes the browser

Code: **package** steps;

**import** org.openqa.selenium.By;

**import** org.openqa.selenium.WebDriver;

**import** org.openqa.selenium.chrome.ChromeDriver;

**import** io.cucumber.java.en.And;

**import** io.cucumber.java.en.Given;

**import** io.cucumber.java.en.Then;

**import** io.cucumber.java.en.When;

**public** **class** TestNinjaTitleFeature {

// In cucmber we dont use Junit Annotations

// we use cucumber annotation:

// @Given , @Then, @When, @And, @But

// In this file we create method for every feature step

**public** **static**  WebDriver *driver*;

@Given("User is on Chrome browser")

**public** **void** OpenBrowser()

{

*driver* = **new** ChromeDriver();

*driver*.manage().deleteAllCookies();

*driver*.manage().window().maximize();

}

@When("User is on Chrome browser users enters the URL")

**public** **void** EnterURL()

{

*driver*.get("https://tutorialsninja.com/demo/");

}

@Then("User gets title of the page")

**public** **void** testpageTitle()

{

String title = *driver*.getTitle();

System.***out***.println(title);

}

@When("User will enter a product in search box")

**public** **void** user\_will\_enter\_a\_product\_in\_search\_box() {

*driver*.findElement(By.*xpath*("//input[@placeholder='Search']")).sendKeys("imac");

}

@When("user click on submit button")

**public** **void** user\_click\_on\_submit\_button() {

*driver*.findElement(By.*xpath*("//button[@class='btn btn-default btn-lg']")).click();

}

@Then("product list should be displayed")

**public** **void** product\_list\_should\_be\_displayed() {

System.***out***.println(*driver*.getTitle());

}

@When("user click on Addto cart link")

**public** **void** user\_click\_on\_addto\_cart\_link() {

*driver*.findElement(By.*xpath*("//a[@title='Shopping Cart']//i[@class='fa fa-shopping-cart']")).click();

}

@Then("user is navigated to Cart page")

**public** **void** user\_is\_navigated\_to\_cart\_page() {

System.***out***.println(*driver*.getTitle());

}

@And("user closes the browser")

**public** **void** teardown()

{

*driver*.close();

}

}

Question 11:- You are given a project to demonstrate the use of Logical AND/OR in a scenario.

package runner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features=

"C:\\Users\\sadiq\\eclipse-workspace\\cocumber\_practise\\src\\test\\java\\features\\hooksdemo1.feature ",

glue = {"steps"}, // packagename where the steps are

plugin= {"pretty","html:target/cucumberreport.html"},

//dryRun=true

// tags="@sanity", // all the scenario is feature file with tagname sanity will get executed

tags="@regression and @search" // and operator// scanerios whicha re having both of thes etags will run

tags="@regression or @sanity"

// tags="@all" // all scnaerios will run

//tags = "not @search" // run all the scenarios except scenario with tag search

tags="@regression"

)

public class hooksdemo1runner {

// we dont write anythign over here.

}

Question 12:-

You are given a project to demonstrate how Cucumber data tables are used.

Feature file

Feature: Test the WikiPage Login on ChromeBrowser

Scenario Outline: Test Creation of Account on WikiPage

Given User is on the MainPage,get title of the page

When User will enter '<username>','<password>'

Then User will enter '<retype>' and '<email>'

Then user will click on Create account button

Examples:

| username | password | retype | email |

| user1 | password1 | password1 | email@gmail.com |

Code:- package steps;

import org.openqa.selenium.By;

import org.openqa.selenium.WebElement;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class exampledemo1steps {

@Given("User is on the MainPage,get title of the page")

public void user\_is\_on\_the\_main\_page\_get\_title\_of\_the\_page() {

String title = BaseHooks.driver.getTitle();

System.out.println(title);

}

@When("User will enter {string},{string}")

public void user\_will\_enter\_username\_password(String user, String passwd) throws InterruptedException {

WebElement e1 = BaseHooks.driver.findElement(By.xpath("//input[@id='wpName2']"));

WebElement e2 = BaseHooks.driver.findElement(By.xpath("//input[@id='wpPassword2']"));

e1.clear();

e1.sendKeys(user);

e2.clear();

e2.sendKeys(passwd);

Thread.sleep(2000);

}

@Then("User will enter {string} and {string}")

public void user\_will\_enter\_emial\_id\_retype\_password(String re, String em) throws InterruptedException {

WebElement e3 = BaseHooks.driver.findElement(By.xpath("//input[@id='wpRetype']"));

WebElement e4 = BaseHooks.driver.findElement(By.xpath("//input[@id='wpEmail']"));

e3.clear();

e3.sendKeys(re);

e4.clear();

e4.sendKeys(em);

Thread.sleep(2000);

}

@Then("user will click on Create account button")

public void user\_will\_click\_on\_create\_account\_button() {

WebElement e5 = BaseHooks.driver.findElement(By.xpath("//button[@id='wpCreateaccount']"));

e5.click();

}

}

Question 13:- You are given a project to demonstrate Cucumber Integration with Extent Report.

IntegratingCucumber with Extent Report

* In POM.Xml please add below maven dependencies.

|  |
| --- |
| <dependency>  <groupId>com.aventstack</groupId>  <artifactId>extentreports</artifactId>  <version>3.0.6</version>  </dependency>  <dependency>  <groupId>com.vimalselvam</groupId>  <artifactId>cucumber-extentsreport</artifactId>  <version>3.0.2</version> |

* Add ***extent-config.xml*** *file under test/resources in your maven project.*

******

* *And add below code inside it.*

|  |
| --- |
| ***<?xml version="1.0" encoding="UTF-8"?>***  ***<extentreports>***  ***<configuration>***  ***<!-- report theme -->***  ***<!-- standard, dark -->***  ***<theme>standard</theme>***    ***<!-- document encoding -->***  ***<!-- defaults to UTF-8 -->***  ***<encoding>UTF-8</encoding>***    ***<!-- protocol for script and stylesheets -->***  ***<!-- defaults to https -->***  ***<protocol>https</protocol>***    ***<!-- title of the document -->***  ***<documentTitle>ExtentReports 2.0</documentTitle>***    ***<!-- report name - displayed at top-nav -->***  ***<reportName></reportName>***    ***<!-- report headline - displayed at top-nav, after reportHeadline -->***  ***<reportHeadline>Automation Report</reportHeadline>***    ***<!-- global date format override -->***  ***<!-- defaults to yyyy-MM-dd -->***  ***<dateFormat>yyyy-MM-dd</dateFormat>***    ***<!-- global time format override -->***  ***<!-- defaults to HH:mm:ss -->***  ***<timeFormat>HH:mm:ss</timeFormat>***    ***<!-- custom javascript -->***  ***<scripts>***  ***<![CDATA[***  ***$(document).ready(function() {***    ***});***  ***]]>***  ***</scripts>***    ***<!-- custom styles -->***  ***<styles>***  ***<![CDATA[***    ***]]>***  ***</styles>***  ***</configuration>***  ***</extentreports>*** |

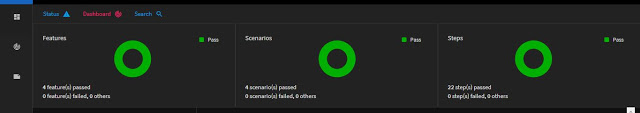
* In the Runner File under plugin add below code.

|  |
| --- |
| Plugin={“com.cucumber.listener.ExtentCucumberFormatter:target/html/ExtentReport.html “} |

* Also add below code in After class of runner file.

|  |
| --- |
| public class TestRunner  {  @AfterClass  public **static** **void** setup()  {  Reporter.loadXMLConfig(new File("src/test/resources/extent-config.xml"));  Reporter.setSystemInfo("User Name", "AJ");  Reporter.setSystemInfo("Application Name", "Test App ");  Reporter.setSystemInfo("Operating System Type", System.getProperty("os.name").toString());  Reporter.setSystemInfo("Environment", "Production");  Reporter.setTestRunnerOutput("Test Execution Cucumber Report");  }  } |

Now, after executing the code extent report would be generated inside /target/html folder and it would be looking something like this



Feature file:-

Feature: Register multiple users on the Rediff account Page

Scenario: Test rediff Register Account Page

Given User Opens chrome Browser

When User enter the rediff account page URL, User captures the title

Then User enter following details to create account

|Ravi|id1|pass1|pass1|email@gmail.com|9891234562|

|John|id2|pass3|pass3|email1@gmail.com|9897234562|

|Marc|id3|pass4|pass4|email2@gmail.com|9890234562|

|hack|id4|pass5|pass5|email3@gmail.com|9896234562|

Then User will close the browser

Steps file:-

package steps;

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 io.cucumber.datatable.DataTable;

import io.cucumber.java.en.Given;

import io.cucumber.java.en.Then;

import io.cucumber.java.en.When;

public class TestRediffAccountPage {

WebDriver driver;

@Given("User Opens chrome Browser")

public void open\_Browser()

{

driver = new ChromeDriver();

driver.manage().window().maximize();

}

@When("User enter the rediff account page URL, User captures the title")

public void get\_title\_enter\_url()

{

driver.get("https://register.rediff.com/register/register.php?FormName=user\_details");

System.out.println("The title of page is: " + driver.getTitle());

}

@Then("User enter following details to create account")

public void create\_account(DataTable datatable) throws InterruptedException

{

WebElement e1 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[1]"));

WebElement e2 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[2]"));

WebElement e3 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[4]"));

WebElement e4 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[5]"));

WebElement e5 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[6]"));

WebElement e6 = driver.findElement(By.xpath("//table[@id='tblcrtac']/descendant::input[13]"));

List<List<String>> userList = datatable.asLists(String.class);

for(List<String> e: userList)

{

e1.clear();

e1.sendKeys(e.get(0));

e2.clear();

e2.sendKeys(e.get(1));

e3.clear();

e3.sendKeys(e.get(2));

e4.clear();

e4.sendKeys(e.get(3));

e5.clear();

e5.sendKeys(e.get(4));

e6.clear();

e6.sendKeys(e.get(5));

Thread.sleep(2000);

}

}

@Then("User will close the browser")

public void close\_browser()

{

driver.close();

}

}

Runnerfile:-

package runner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features=

"C:\\Users\\sadiq\\eclipse-workspace\\extentreportincocumber\\src\\test\\java\\feature\\rediffdemo1.feature",

glue = {"steps"}, // packagename where the steps are

plugin= {"pretty","html:target/cucumberreport.html",

"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",

"timeline:test-output-thread/"}

)

public class TestRunner {

}

Extent properties:- extent.reporter.spark.start=true

extent.reporter.spark.out=test-output/SparkReport/Spark.html

extent.reporter.spark.config=src/test/resources/extent-config.xml

extent.reporter.spark.out=test-output/SparkReport/

screenshot.dir=test-output/

screenshot.rel.path=../

extent.reporter.pdf.start=true

extent.reporter.pdf.out=testoutput1/PdfReport/ExtentPdf.pdf

#basefolder.name=reports

#basefolder.datetimepattern=d-MMM-YY HH-mm-ss

extent.reporter.spark.vieworder=dashboard,test,category,exception,author,device,log

systeminfo.os=Windows

systeminfo.user=

systeminfo.build=1.1

systeminfo.AppName=extentreport

Extent-config.xml:-

<?xml version="1.0" encoding="UTF-8"?>

<extentreports>

<configuration>

<!-- report theme -->

<!-- standard, dark -->

<theme>dark</theme>

<!-- document encoding -->

<!-- defaults to UTF-8 -->

<encoding>UTF-8</encoding>

<!-- protocol for script and stylesheets -->

<!-- defaults to https -->

<protocol>http</protocol>

<!-- title of the document -->

<documentTitle>Extent</documentTitle>

<!-- report name - displayed at top-nav -->

<reportName>Grasshopper Report</reportName>

<!-- location of charts in the test view -->

<!-- top, bottom -->

<testViewChartLocation>bottom</testViewChartLocation>

<!-- custom javascript -->

<scripts>

<![CDATA[

$(document).ready(function() {

});

]]>

</scripts>

<!-- custom styles -->

<styles>

<![CDATA[

]]>

</styles>

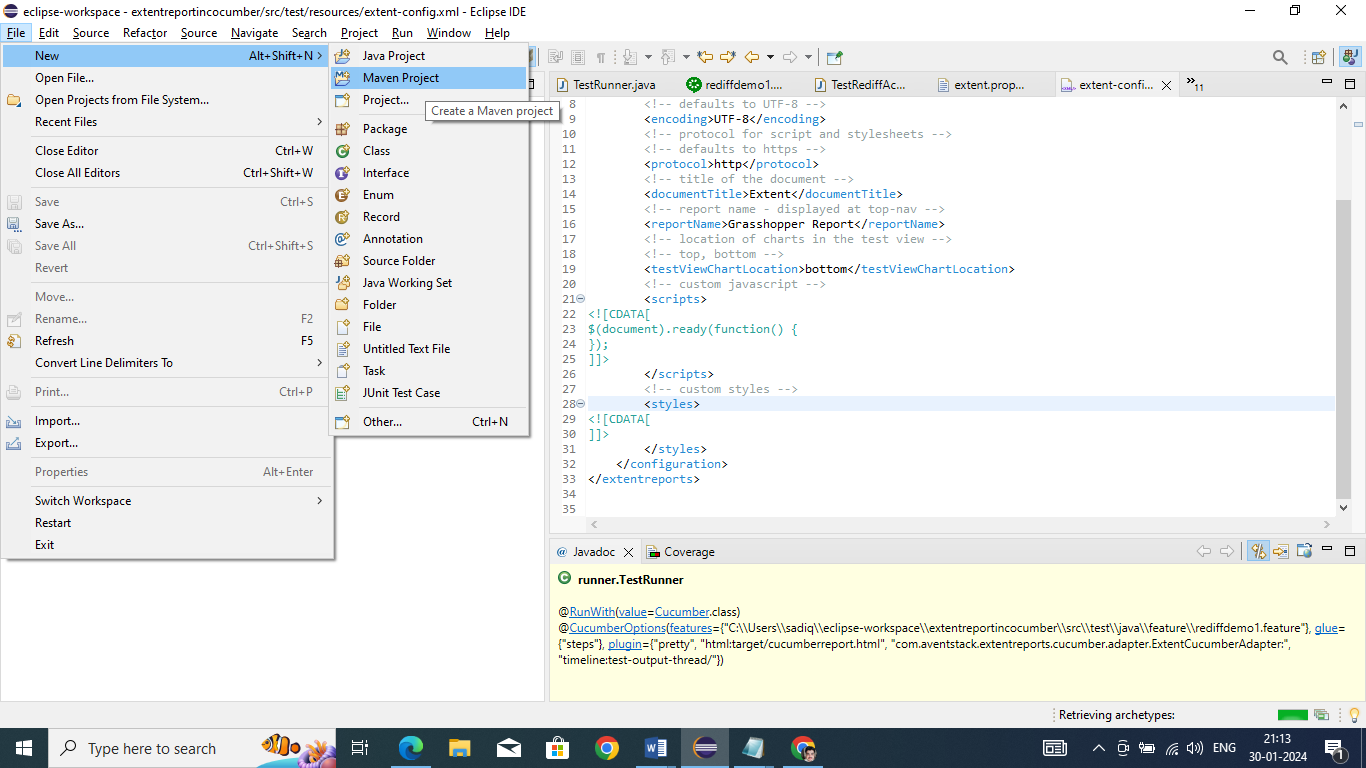
</configuration>

</extentreports>

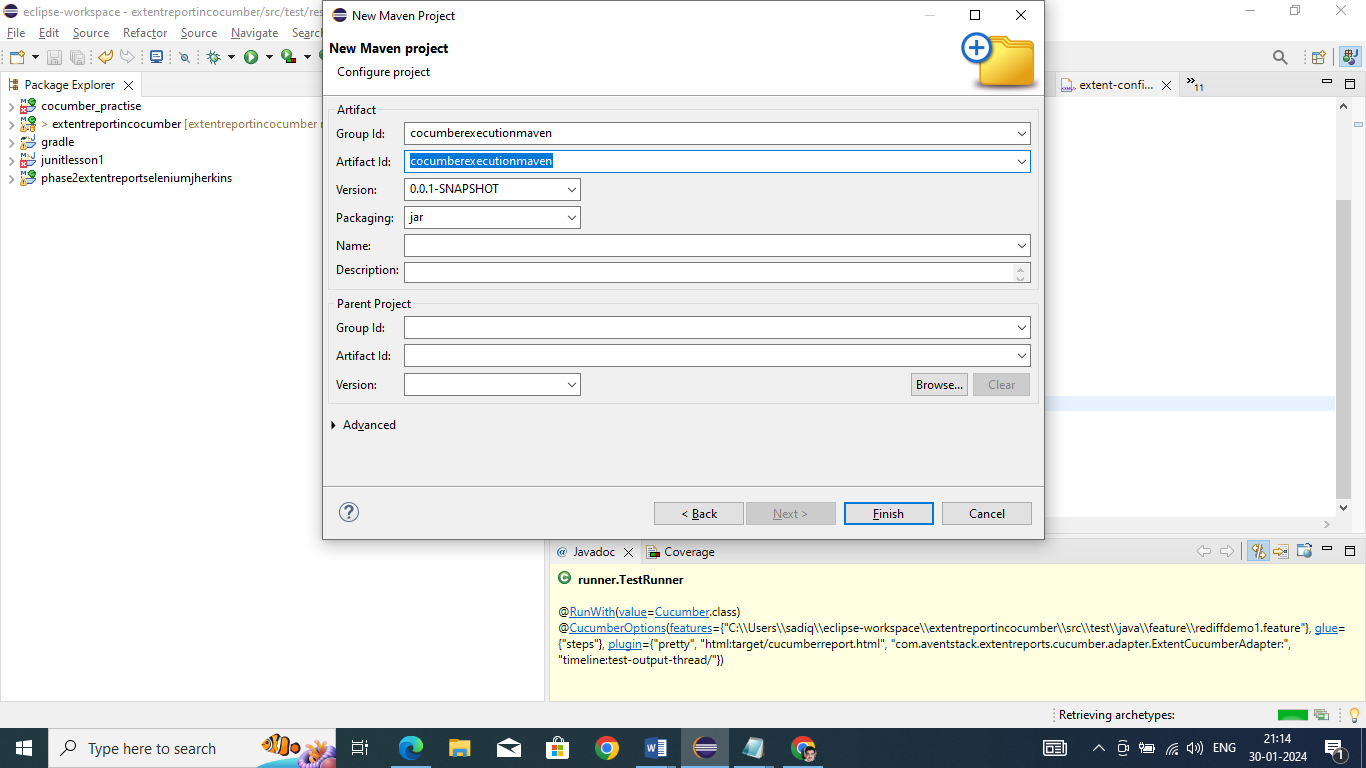
**Question 14** You are given a project to demonstrate complete Cucumber Test Execution with Maven on a local machine

**:** Executing Cucumber Tests with Maven on Local Machine

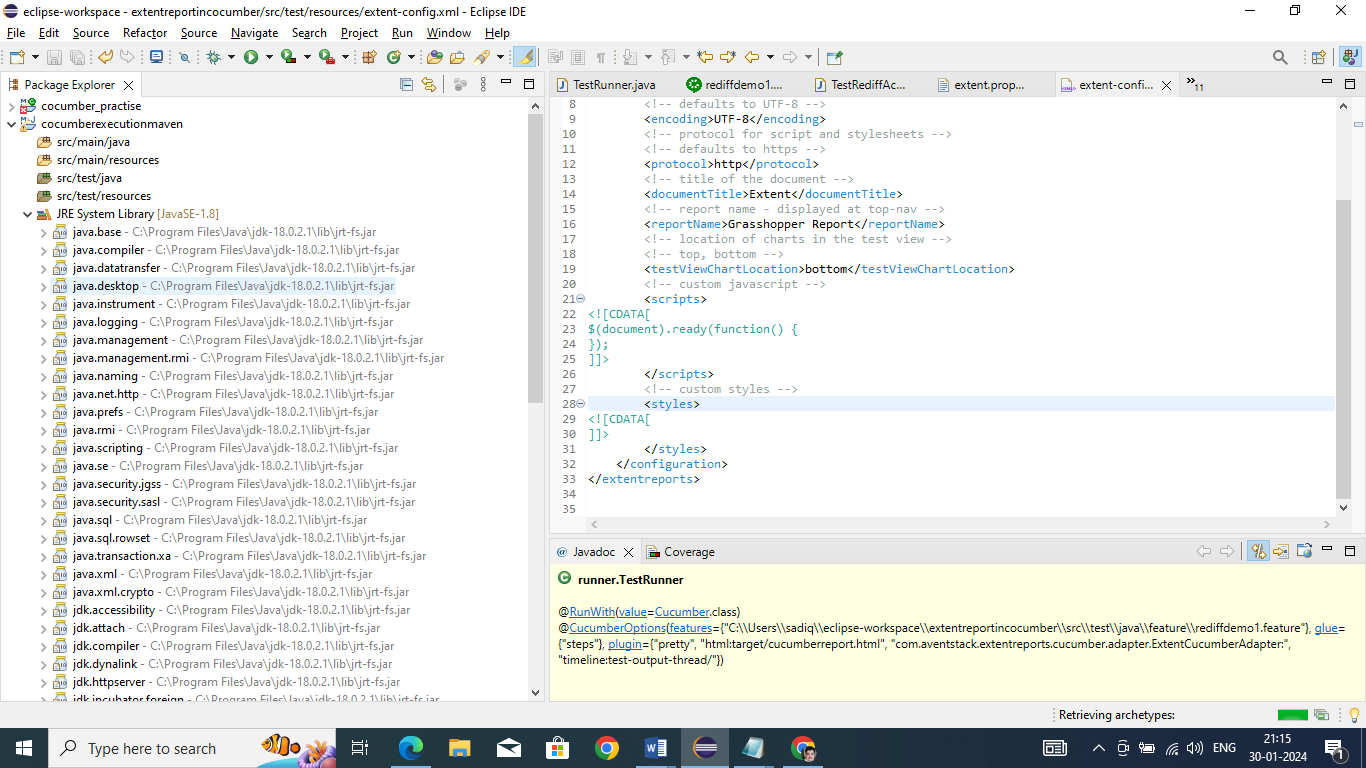
* To create a Maven Project in Eclipse, click on **New** *→* **Project** *→* **In the wizard**, select **Maven Project**.



* On the new Maven Project pop-up, select the checkbox to create your project at the default location OR you can also browse and set a new location of your choice. Click on Next to proceed.
* In the next screen, you will have to mention a Group ID and Artifact ID of your own choice; this is the name of your Maven project. Once you click the Finish button, a Maven project will be created in Eclipse.



* Now, in order to build a Selenium-Cucumber framework, we need to add a dependency for Selenium and Cucumber in pom.xml.
* Copy the dependency tag for the following from the Maven Repository.
  + Selenium Web driver
  + Cucumber-Core jar
  + Cucumber-Java jar
  + Cucumber-TestNG jar
* Make sure to update the project after adding dependencies to pom.xml; you can do that by right clicking *Project → Maven → Update Project*. Once you update the project, you will see that many JAR files are added to the Maven Dependencies folder in your project.



* To proceed with Cucumber implementation, you need to create three packages to store the feature files, step definition code, and test runner code. Let us create three packages: the features, seleniumgluecode, and runner. To create a new package in src/test/java, right Click the folder-> New-> Package.
* Now create the feature file in the Feature package. Right click ->New ->File->Enter name test.feature.
* Create a class test.java to write the gluecode for the features written. Right click seleniumgluecode->New ->Class->enter name as test and save.
* To run the feature files and their respective code, we need to writed a TestNG runner class. Right click **runner** ->**New**-> **Class**->enter name as **testrunner**.

Feature file

Feature: Register multiple users on the Rediff account Page

Scenario: Test rediff Register Account Page

Given User Opens chrome Browser

When User enter the rediff account page URL, User captures the title

Then User enter following details to create account

|Ravi|id1|pass1|pass1|email@gmail.com|9891234562|

|John|id2|pass3|pass3|email1@gmail.com|9897234562|

|Marc|id3|pass4|pass4|email2@gmail.com|9890234562|

|hack|id4|pass5|pass5|email3@gmail.com|9896234562|

Then User will close the browser

Runner file

package runner;

import org.junit.runner.RunWith;

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)

@CucumberOptions(

features=

"C:\\Users\\sadiq\\eclipse-workspace\\extentreportincocumber\\src\\test\\java\\feature\\rediffdemo1.feature",

glue = {"steps"}, // packagename where the steps are

plugin= {"pretty","html:target/cucumberreport.html",

"com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:",

"timeline:test-output-thread/"}

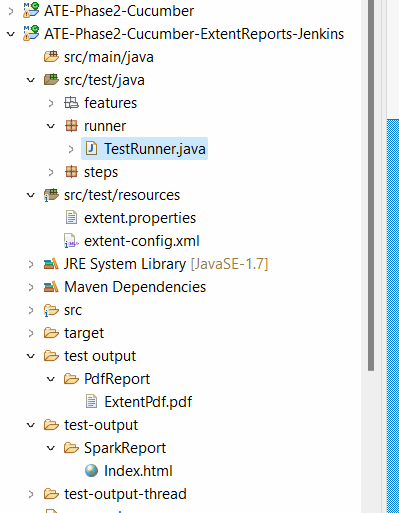
)

public class TestRunner {

}

Execute the runner file.

Refresh the project, you will PDF and spark extent report



How to Run Cucumber Runner file using Maven

================================================

Go to your project→ Open the pom.xml file → scroll down to <build> tag section

Go to surefire plugin and add the configuration tag details as highlighted below

<plugin>

         <artifactId>maven-surefire-plugin</artifactId>

         <version>2.22.1</version>

**<configuration>**

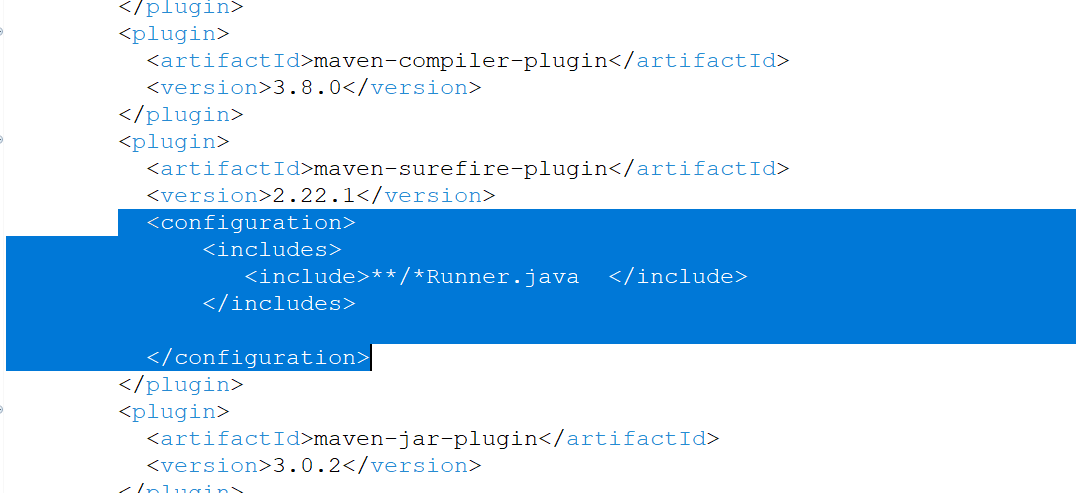
**<includes>**

**<include>\*\*/\*Runner.java  </include>**

**</includes>**

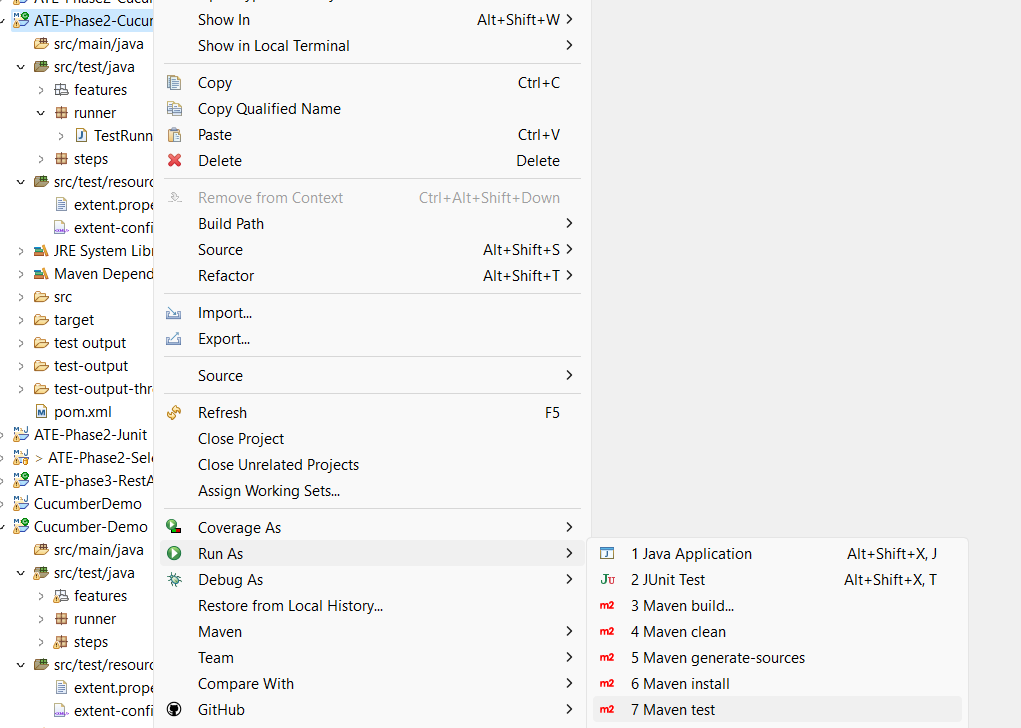
**</configuration>**

       </plugin>



Save the Pom.xml file.

Run the project using maven → rightclick on the project → Runas → select maven test



Question 15:

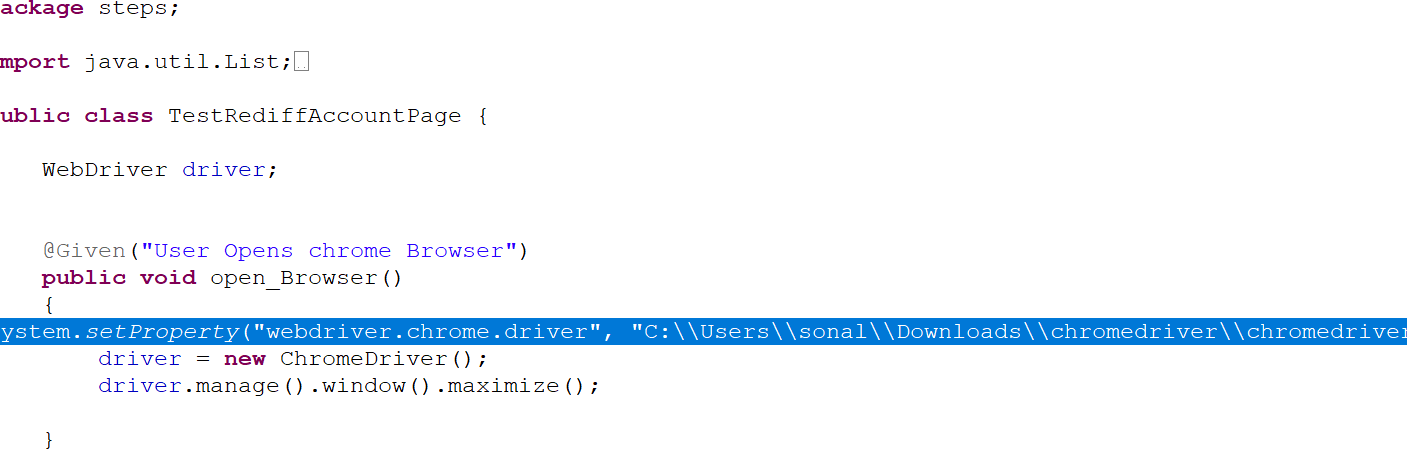
Download chromedriver.exe from selenium webpage

<https://googlechromelabs.github.io/chrome-for-testing/>

And add the below line in the step definition file

System.*setProperty*("webdriver.chrome.driver", "C:\\Users\\sadiq\\Downloads\\chromedriver\\chromedriver-win64\\chromedriver.exe");

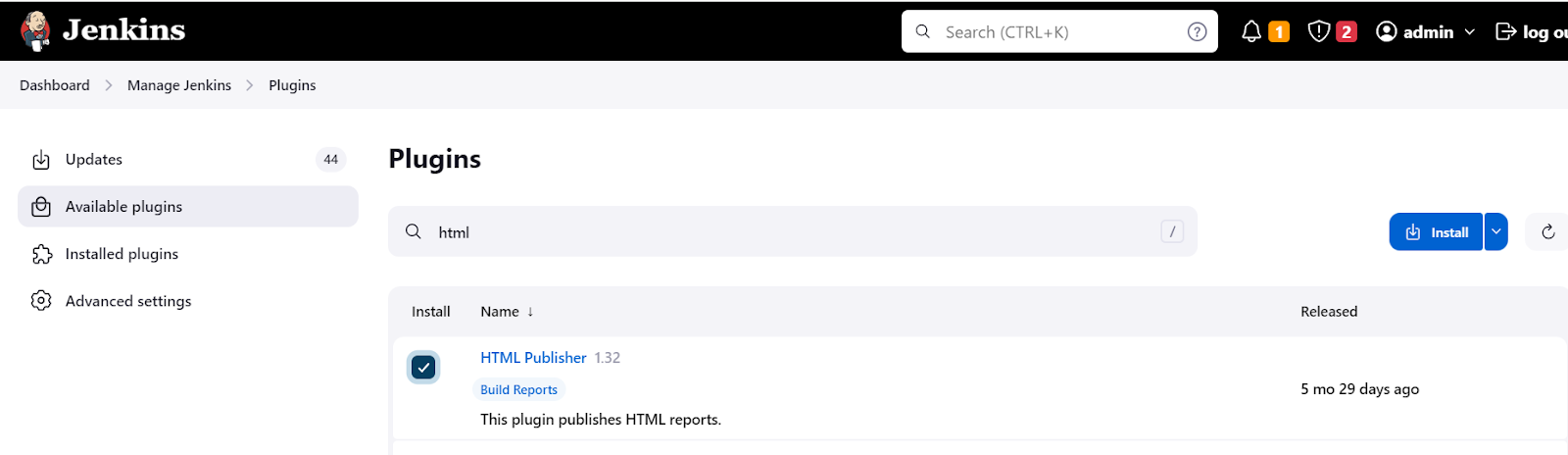
Above give the path of chromedriver exe in your downloads folder.



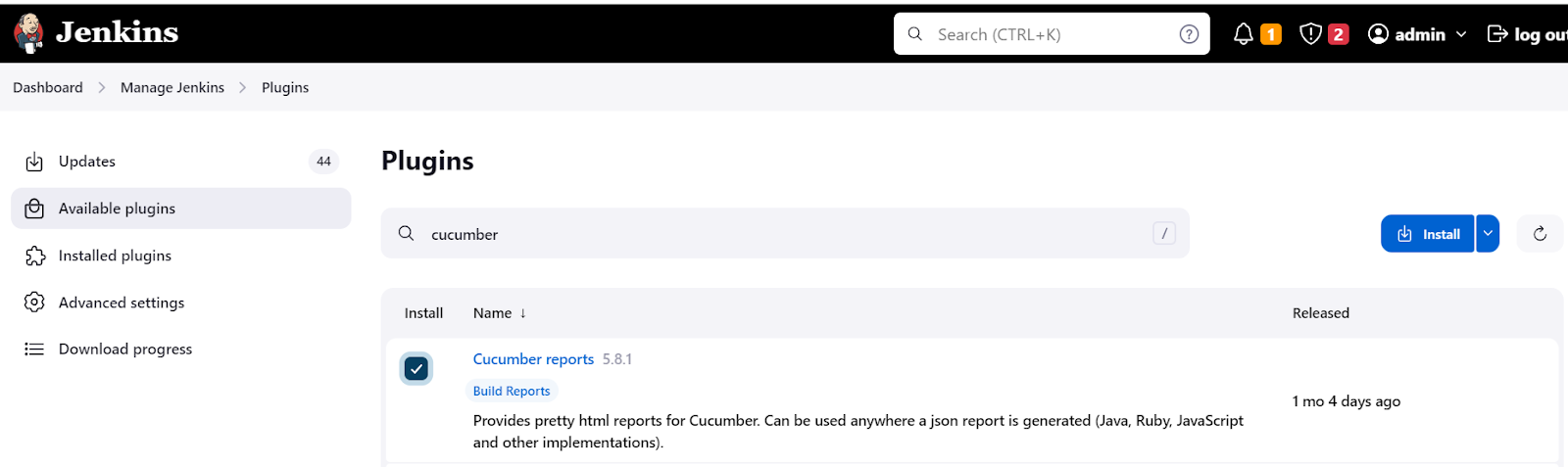
Now push the project to github and run using jenkins job.

In jenkins → manage jenkins→ Plugins→ available plugin

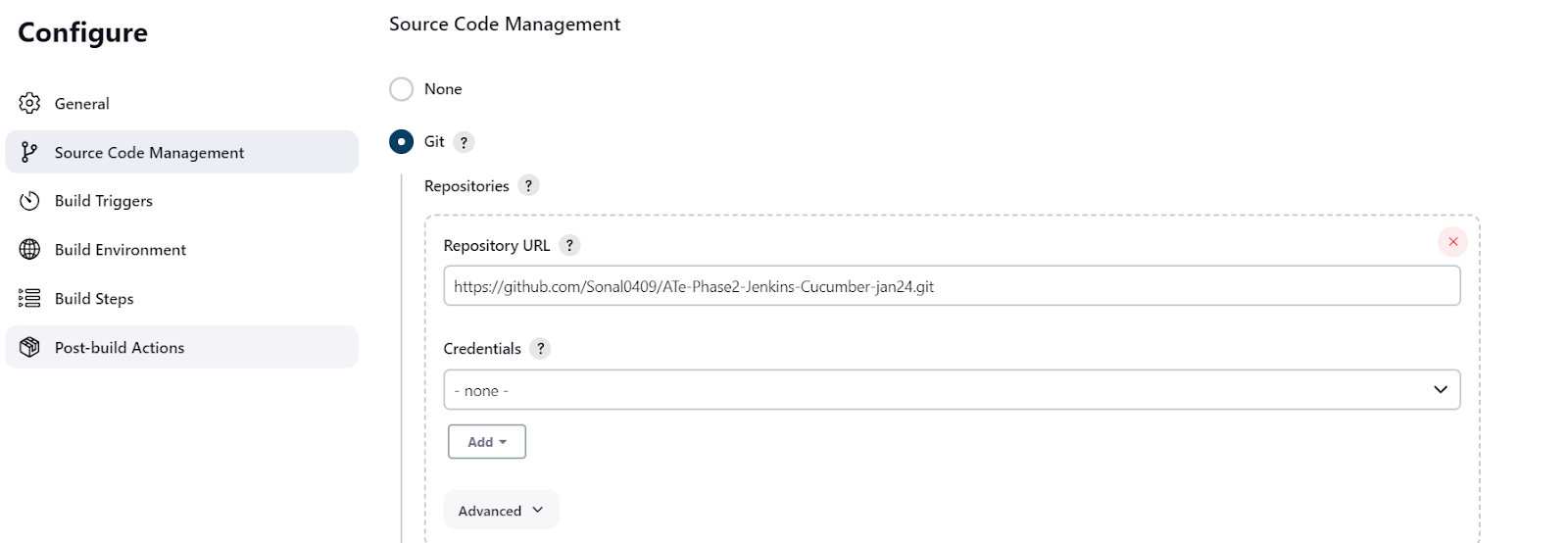
Search for HTML → Install it

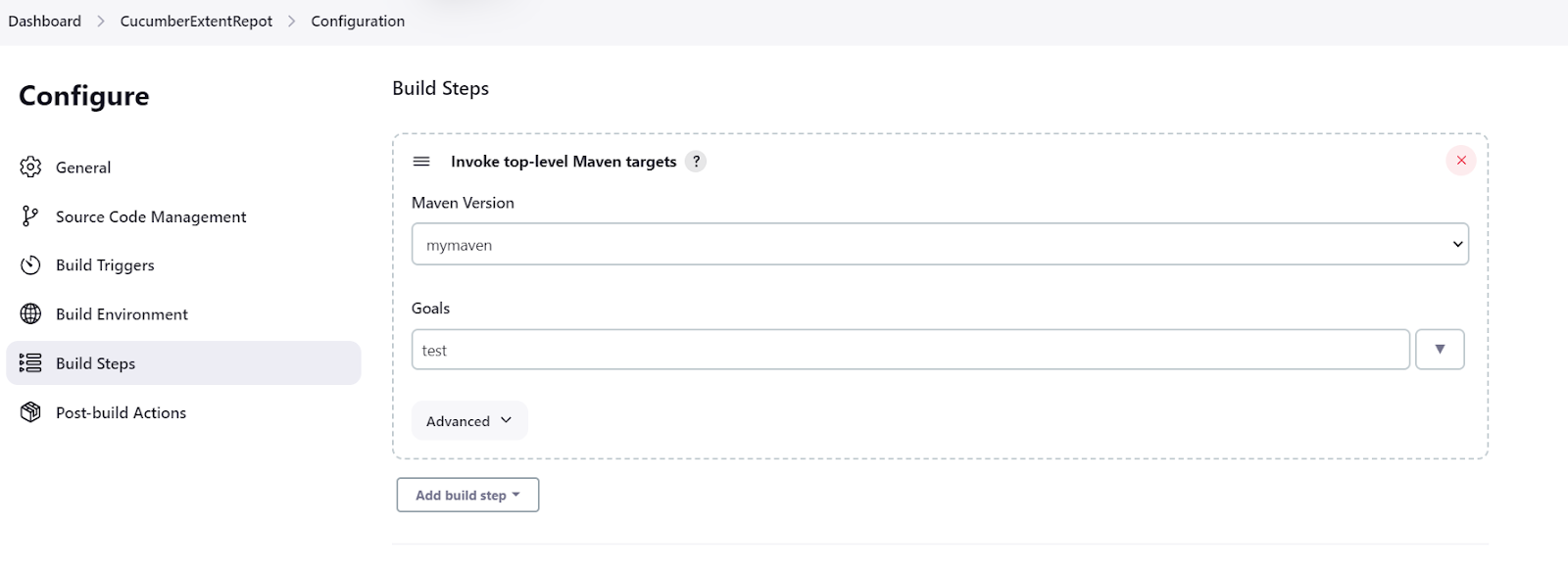


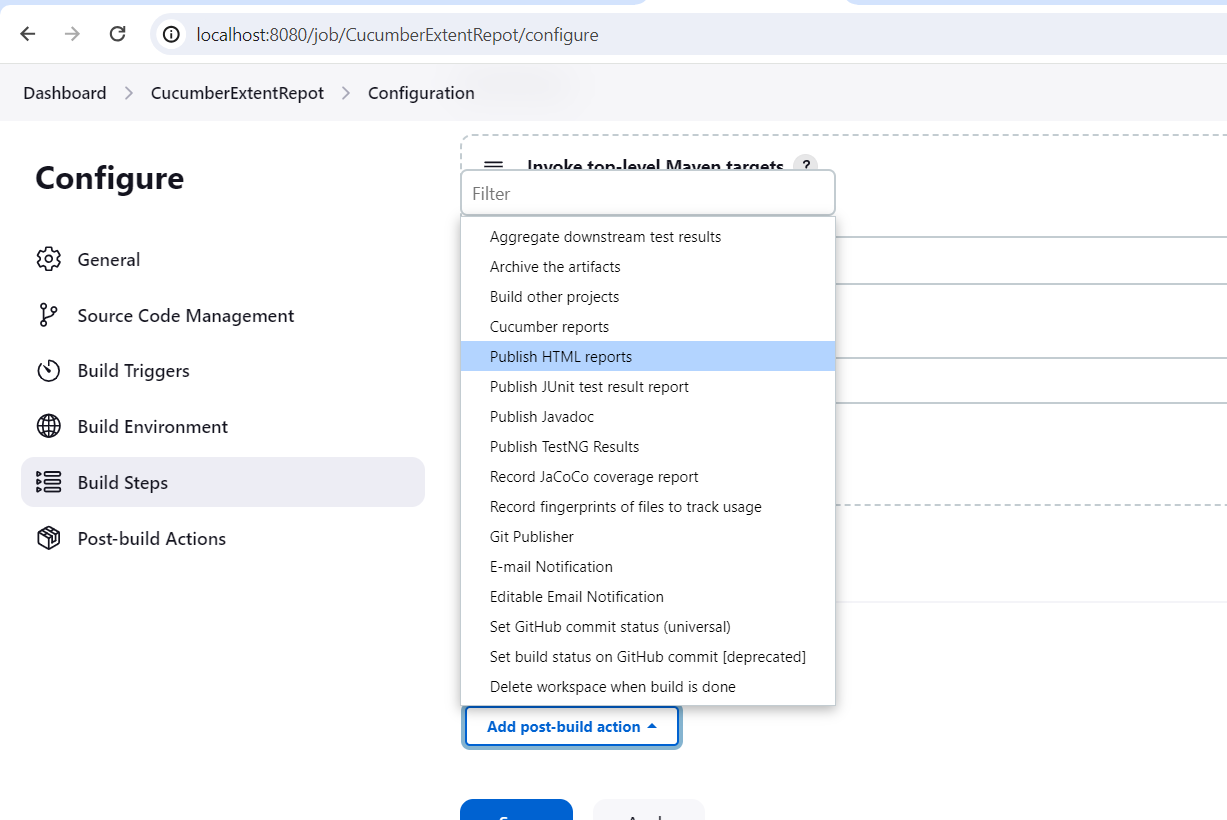
Also download cucumber plugin



Go to your job -> click on configure

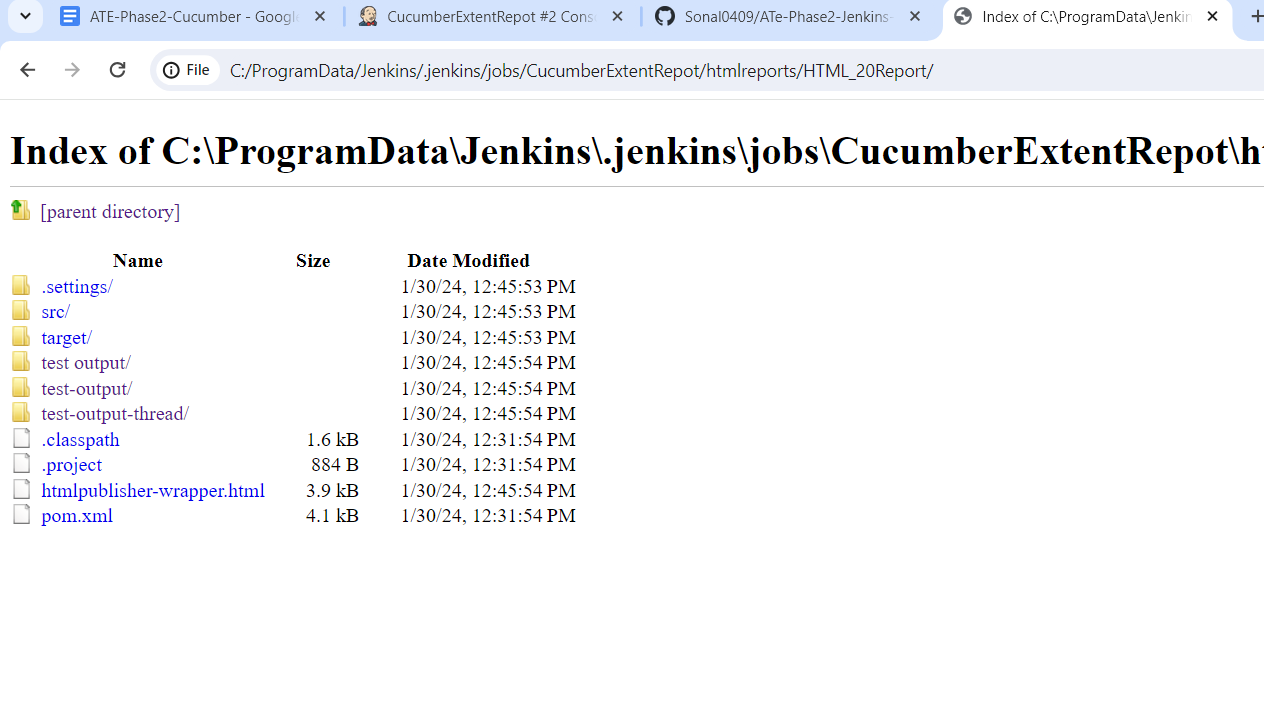


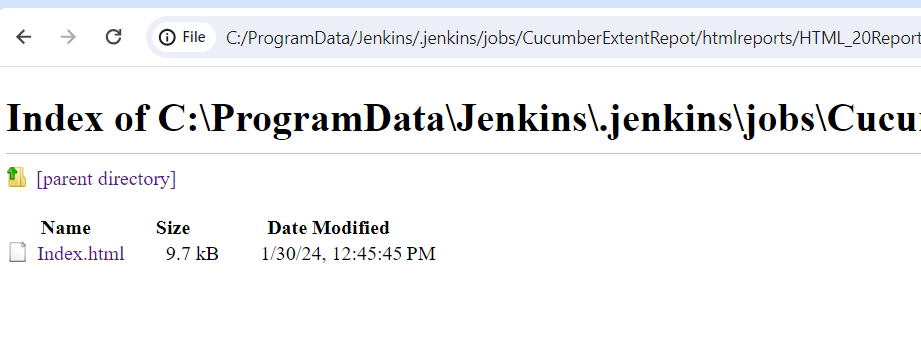




Save the job and run it. You will find reports in the path given in console as below









Note: Add these dependency if you get error with compiler plugin . Other wise not required.

<dependency> <groupId>org.projectlombok</groupId> <artifactId>lombok</artifactId> <version>1.18.24</version> <scope>provided</scope> </dependency>