Framework Design

1. Create a Maven project with simple archetype
2. Create the framework folder structure, by creating the necessary packages
3. Under src/test/java – base, listeners, rough, testCases, utilities
4. Under src/test/resources – excel, executables, logs, properties, runner.
5. Now update the POM.xml file with dependencies – Selenium-Java, extent-report, log4J, javax.mail,

How to implement extent report in Selenium TDD framework.

1. Download extent report latest maven dependency

```
<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->-
<dependency>
  <groupId>com.aventstack</groupId>
  <artifactId>extentreports</artifactId>
  <version>5.1.2</version>
</dependency>
```

2. package SeleniumSessions;

```
import com.aventstack.extentreports.ExtentReports;
import com.aventstack.extentreports.ExtentTest;
import com.aventstack.extentreports.reporter.ExtentSparkReporter;
import com.aventstack.extentreports.reporter.configuration.Theme;
import org.testng.annotations.AfterTest;
import org.testng.annotations.BeforeTest;
import org.testng.annotations.Test;

import static com.aventstack.extentreports.reporter.ExtentSparkReporter.*;

public class TestExtentReports {

  //note- test pass or fail status should be handled from extent
  // if its from testng then it will not be displayed in extent report

  public ExtentSparkReporter htmlReporter; //create html file and add configurations
  public ExtentReports extent; //attaching reporter, creating tests, logs, etc

  public ExtentTest test; // maintaining the test cases, add log status, etc


  @BeforeTest
  public void setReport(){

    htmlReporter = new ExtentSparkReporter("./ExtentReports/extent.html");
```

```
        htmlReporter.config().setEncoding("utf-8");
        htmlReporter.config().setDocumentTitle("Automation Test Report");
        htmlReporter.config().setReportName("Functional Test Report");
        htmlReporter.config().setTheme(Theme.STANDARD);
        htmlReporter.config().setTimeStampFormat("EEEE, MMMM dd, yyyy, hh:mm a
'('zzz')'");
        htmlReporter.config().setTimelineEnabled(true);

        extent = new ExtentReports();
        extent.attachReporter(htmlReporter);

        extent.setSystemInfo("Hostname", "localhost");
        extent.setSystemInfo("Browser", "Chrome");
        extent.setSystemInfo("Environment", "QA");
        extent.setSystemInfo("Tester", "Rahul");


    }

    @AfterTest
    public void endReport(){
        extent.flush();
    }

    @Test
    public void createTest(){
      /**
       * Creates a test in the ExtentReports instance and logs the steps of the test.
       */
        test = extent.createTest("Login Test"); // Create a test named "Login Test" in the
ExtentReports instance

        test.pass("URL is opened"); // Log that the URL is opened
        test.pass("Title is verified"); // Log that the title is verified
        test.pass("Login test"); // Log that the login test is completed

    }

}
```

Cucumber BDD Framework Design

Create maven project in eclipse

update POM.xml file

cucumber-java

cucumber-TestNG

Selenium

install cucumber eclipse plugin

restart eclipse after plugin installation

Right click on the project > configure > convert to cucumber project

cucumber green symbol will be displayed in project

create feature file and keep it under src test resources as feature files doesn't have any code, its plain english using gherkin language

create package features

create file login.feature

Feature: Login Feature

In order to perform successful login

As a user

I have to enter correct username and password

Scenario: Login to the Application

Given user navigates to the application

When user validates the home page title

Then user enters username

And user enters password

And user clicks on SignIn button

right click and select pretty format

yellow highlight as glue code is not there

run the feature file , step def methods will be shown in console

To Run - Run Confifuration > select feature option > enter feature file path

Create steps package under src/test/java

under this create a new java class > loginSteps.java

paste the auto suggested glue code from console here.

import the relevant packages

file > maven > update project, yellow sign will go

Again run the feature file and check

you will see green sign for each step once feature file is mapped to steps definitions

CTRL+click on feature steps will navigate to steps

code written inside steps will have selenium/ api/ appium any code will run

Now create the Runner file (java file) to run our feature file

Under src/test/java create a new package called runner

Create java class RunLogin.java

```
package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;



@CucumberOptions(features="src/test/resources/features/login.feature",
        glue= {"steps"}

                )
public class RunLogin extends AbstractTestNGCucumberTests {




}
```

run the runner class as TestNG

After the runner class is implemented lets implement cucumber report in our framework

modified runner after implementing plugin

```
package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;



@CucumberOptions(features="src/test/resources/features/login.feature",
        glue= {"steps"},
```

```
        plugin= {"html:target/cucumber-reports/cucumber-html-report.html"}

                )
public class RunLogin extends AbstractTestNGCucumberTests {




}
```

run the runner class and check for the report in Target folder after refreshing the project

extent report implementation from tech.grasshopper

search for - extentreports-cucumber 7 Adapter

Add the latest maven dependency in pom.xml file

add extent report latest version dependency also in pom.xml file

Now add extentreports-cucumber 7 Adapter as plugin in your runner file

https://extentreports.com/docs/versions/4/java/cucumber4.html

```
package runner;

import io.cucumber.testng.AbstractTestNGCucumberTests;
import io.cucumber.testng.CucumberOptions;


@CucumberOptions(features="src/test/resources/features/login.feature",
        glue= {"steps"},
```

```
        plugin= {"html:target/cucumber-reports/cucumber-html-report.html",
                "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}


                )
public class RunLogin extends AbstractTestNGCucumberTests {




}
```

Once the adapter plugin is configured in the runner class, under src/test/resources/ create extent.properties file

so basically the extent plugin is going to refer this extent.properties file.

So report activation write

Note- There are two files extent.properties and extent-config.xml keep these file under src/test/resources/ to make it work

extent-config.xml file will be referred from extent.properties file

```
# For report Activation
extent.reporter.spark.start=true
extent.reporter.spark.out=MyReports/Spark.html
extent.reporter.spark.config=src/test/resources/extent-config.xml
```

sample extent-config.xml

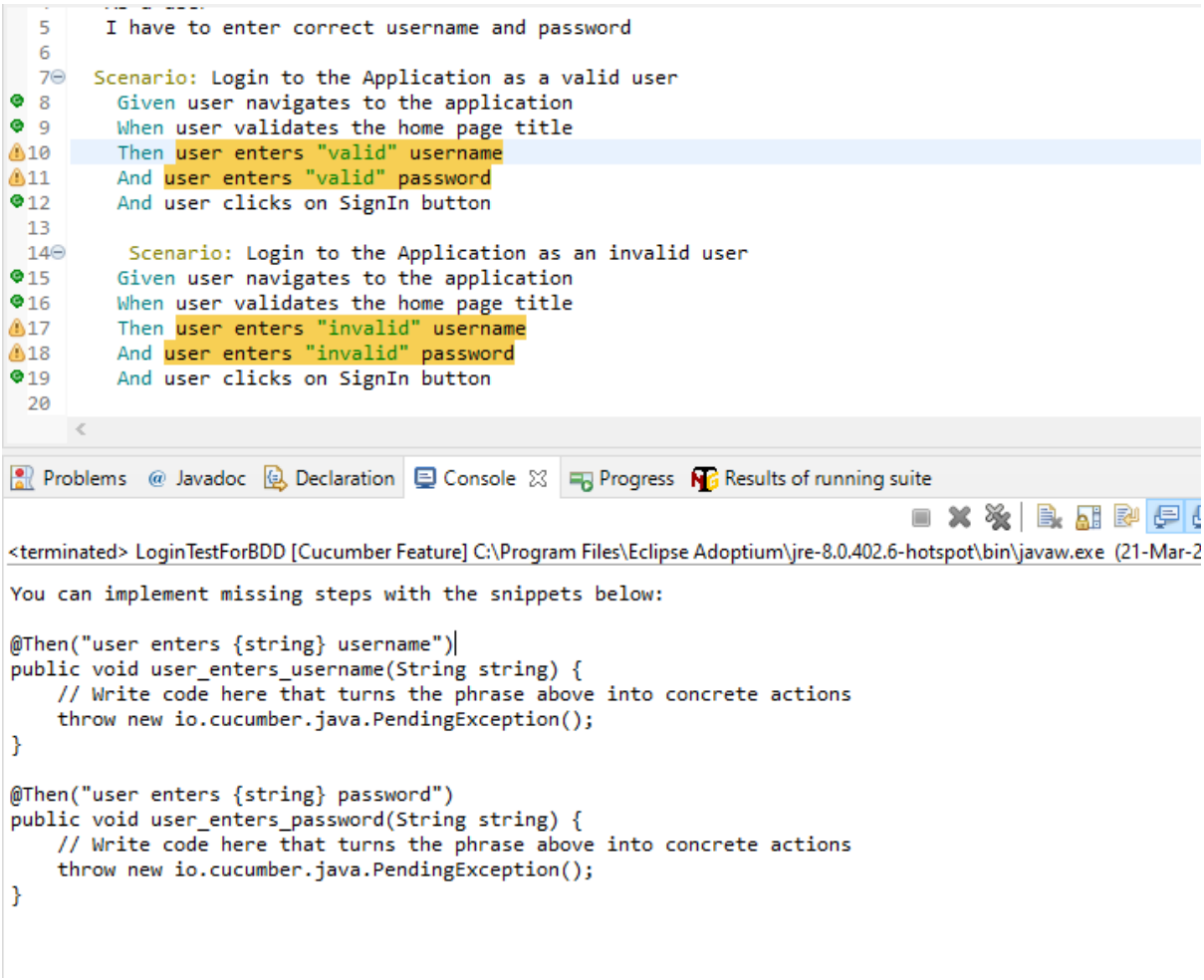
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<extentreports>
  <configuration>
    <!-- report theme -->
    <!-- standard, dark -->
    <theme>standard</theme>

    <!-- enables timeline -->
    <!-- defaults to true -->
    <enableTimeline>true</enableTimeline>

    <!-- 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>Extent Framework</documentTitle>

    <!-- report name - displayed at top-nav -->
    <reportName>Build 1</reportName>

    <!-- custom javascript -->
    <scripts>
      <![CDATA[
        $(document).ready(function() {

        });
      ]]>
```

```
    </scripts>

    <!-- custom styles -->
    <styles>
      <![CDATA[

      ]]>
    </styles>
  </configuration>
</extentreports>
```

https://extentreports.com/docs/versions/4/java/spark-reporter.html

Adding multiple scenarios to a feature file




```java
@Then("user enters {string} username")
public void user_enters_username(String username) {

    System.out.println("@Then---- user enters "+ username+"username");

}

@Then("user enters {string} password")
public void user_enters_password(String password) {
    System.out.println("@Then---- user enters"+password+"password");

}
```

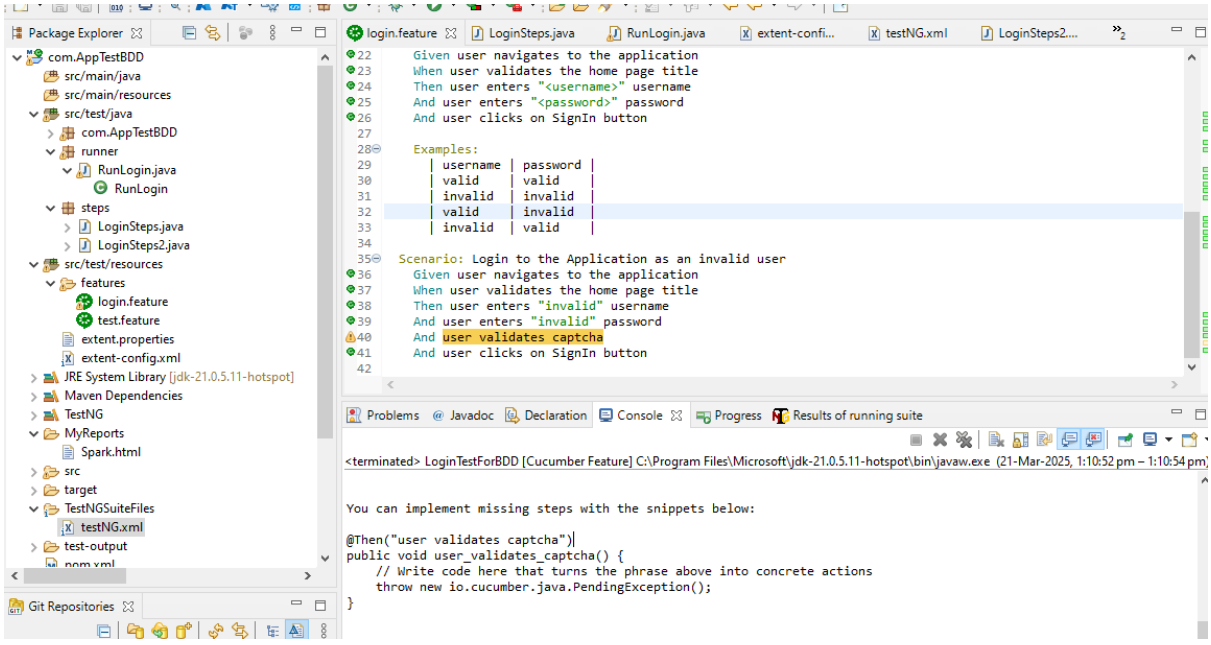
No of {string} counts = no of parameters in the method.

```
  Scenario Outline: Login to the Application with different combinations
    Given user navigates to the application
    When user validates the home page title
    Then user enters "<username>" username
    And user enters "<password>" password
    And user clicks on SignIn button

    Examples:
      | username | password |
      | valid    | valid    |
      | invalid  | invalid  |
      | valid    | invalid  |
      | invalid  | valid    |
```

```
@Then("user enters {string} username")
public void user_enters_username(String username) {

    System.out.println("@Then---- user enters "+ username+"username");

}

@Then("user enters {string} password")
public void user_enters_password(String password) {
    System.out.println("@Then---- user enters"+password+"password");

}
```

Adding multiple features and Step Definitions

Under the steps package we can add multiple java classes for steps definitions and all the steps will be visible throughout the features.



Background

loginBackgroundExample.feature | testNGStaging.xml

```
1  Feature: Login Feature
2
3     In order to perform successful login
4     As a user
5     I have to enter correct username and password
6
7    Background:
8      Given user navigates to the application
9      When user validates the home page title
10
11   Scenario: Login to the Application as a valid user
12     Then user enters "valid" username
13     And user enters "valid" password
14     And user clicks on SignIn button
15
16   Scenario: Login to the Application as an invalid user
17     Then user enters "invalid" username
18     And user enters "invalid" password
19     And user clicks on SignIn button
20
```

Problems | Javadoc | Declaration | Console | Progress | Results of running class RunL

```
<terminated> RunLoginStaging [TestNG] C:\Program Files\Microsoft\jdk-21.0.5.11-hotspot\bin\javaw.exe (21-
[RemoteTestNG] detected TestNG version 7.4.0
@GIVEN---- user navigates to the application
@When---- user validates the home page title
@Then---- user enters validusername
@Then---- user entersvalidpassword
@Then---- user clicks on SignIn button
@GIVEN---- user navigates to the application
@When---- user validates the home page title
@Then---- user enters invalidusername
@Then---- user entersinvalidpassword
@Then---- user clicks on SignIn button
PASSED: runScenario("Login to the Application as a valid user", "Login Feature")
```

Cucumber Tags

Used when we want to run some set of selected scenarios smoke,sanity, modulewise

To execute limited features, we can tag our features and scenarios.

loginBackgroundExample.feature | testNGStaging.xml | *login.feature

```
@regression
Feature: Login Feature

  In order to perform successful login
  As a user
  I have to enter correct username and password

Scenario: Login to the Application as a valid user
  Given user navigates to the application
  When user validates the home page title
  Then user enters "valid" username
  And user enters "valid" password
  And user clicks on SignIn button

@regression
Scenario: Login to the Application as an invalid user
  Given user navigates to the application
  When user validates the home page title
  Then user enters "invalid" username
  And user enters "invalid" password
  And user clicks on SignIn button

@smoke
Scenario Outline: Login to the Application with different combinations
  Given user navigates to the application
  When user validates the home page title
  Then user enters "<username>" username
  And user enters "<password>" password
  And user clicks on SignIn button

  Examples:
    | username | password |
    | valid    | valid    |
    | invalid  | invalid  |
```

To make this work under @cucumberOptions add tags=”regression”

If you don’t mention any tags then we can run all the features in features folder in case if we don’t want to run any specific feature file.

loginBackgroundExample.feature | login.feature | RunLogin.java | testNG.xml

```java
package runner;

import com.aventstack.extentreports.ExtentReports;


@CucumberOptions(features="src/test/resources/features",
                 glue= {"steps"},
                 tags="@staging",
                 plugin= {"html:target/cucumber-reports/cucumber-html-report.html",
                         "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}

        )
public class RunLogin extends AbstractTestNGCucumberTests {


}
```

loginBackgroundExample.feature | login.feature | RunLogin.java | testNG.xml

```gherkin
@staging
Feature: Login Feature staging

  In order to perform successful login
  As a user
  I have to enter correct username and password


  Background:
    Given user navigates to the application
    When user validates the home page title

  @staging
  Scenario: Login to the Application as a valid user on staging
    Then user enters "valid" username
    And user enters "valid" password
    And user clicks on SignIn button

  @staging
  Scenario: Login to the Application as an invalid user on staging
    Then user enters "invalid" username
    And user enters "invalid" password
    And user clicks on SignIn button

```

Problems | Javadoc | Declaration | Console | Progress | Results of running class RunLogin

```
<terminated> RunLogin [TestNG] C:\Program Files\Microsoft\jdk-21.0.5.11-hotspot\bin\javaw.exe  (21-Mar-2025, 2:45:53 pm – 2:45:57 pm)
@Then---- user entersinvalidpassword
@Then---- user clicks on SignIn button
PASSED: runScenario("Login to the Application as an invalid user on staging", "Login Feature staging")
        Runs Cucumber Scenarios
PASSED: runScenario("Login to the Application as a valid user on staging", "Login Feature staging")
        Runs Cucumber Scenarios

===============================================
    Default test
```

loginBackgroundExample.feature | login.feature | RunLogin.java | testNG.xml | RunLoginStaging.java

```java
package runner;

import com.aventstack.extentreports.ExtentReports;


@CucumberOptions(features="src/test/resources/features",
                 glue= {"steps"},
                 tags="@production and not @staging",
                 plugin= {"html:target/cucumber-reports/cucumber-html-report.html",
                         "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}

        )
public class RunLoginStaging extends AbstractTestNGCucumberTests {



}
```

loginBackgroundExample.feature | login.feature | RunLogin.java | testNG.xml | RunLoginStaging.java

```gherkin
@production
Feature: Login Feature production

   In order to perform successful login
   As a user
   I have to enter correct username and password

  Scenario: Login to the Application as a valid user on production
    Given user navigates to the application
    When user validates the home page title
    Then user enters "valid" username
    And user enters "valid" password
    And user clicks on SignIn button

  Scenario: Login to the Application as an invalid user on production
    Given user navigates to the application
    When user validates the home page title
    Then user enters "invalid" username
    And user enters "invalid" password
    And user clicks on SignIn button

  Scenario Outline: Login to the Application with different combinations on production
    Given user navigates to the application
    When user validates the home page title
    Then user enters "<username>" username
    And user enters "<password>" password
    And user clicks on SignIn button

    Examples:
      | username | password |
      | valid    | valid    |
      | invalid  | invalid  |
      | valid    | invalid  |
      | invalid  | valid    |
```

Tags=”@production or @staging or @sanity” run both

Tags=”@production or @staging and not @sanity”

Cucumber Hooks

Hooks is similar to @Before and @After annotation in TestNG and junit

Create hooks class in your steps package

loginBackgroundE... | login.feature | RunLogin.java | testNG.xml | RunLoginStaging.ja... | Hooks.java

```java
package steps;

import io.cucumber.java.After;
import io.cucumber.java.Before;

public class Hooks {

    //works for each scenario

    @Before
    public void setUp() {

        System.out.println("@Before Hook---- Launching the browser");
    }

    @After
    public void tearDown() {

        System.out.println("@After Hook---- quitting the browser");
    }
}
```

```
[RemoteTestNG] detected TestNG version 7.4.0
@Before Hook---- Launching the browser
@GIVEN---- user navigates to the application
@When---- user validates the home page title
@Then---- user enters validusername
@Then---- user entersvalidpassword
@Then---- user clicks on SignIn button
@After Hook---- quitting the browser
@Before Hook---- Launching the browser
@GIVEN---- user navigates to the application
@When---- user validates the home page title
@Then---- user enters invalidusername
@Then---- user entersinvalidpassword
@Then---- user clicks on SignIn button
@After Hook---- quitting the browser
PASSED: runScenario("Login to the Application as a valid user on staging", "Login Feature staging")
        Runs Cucumber Scenarios
PASSED: runScenario("Login to the Application as an invalid user on staging", "Login Feature staging")
        Runs Cucumber Scenarios

===============================================
    Default test
    Tests run: 1, Failures: 0, Skips: 0
===============================================


===============================================
```

Note- @Before and @After – work before and after each scenario in a feature file.

loginBackgroundE... | login.feature | RunLogin.java | testNG.xml | RunLoginStaging.j... | *Hooks.java

```java
package steps;
import io.cucumber.java.After;
import io.cucumber.java.AfterAll;
import io.cucumber.java.Before;
import io.cucumber.java.BeforeAll;

public class Hooks {

    @BeforeAll
    public void before_All() {

        System.out.println("@BeforeAll --- Open DB connection ");

    }

    @AfterAll
    public void after_All() {
        System.out.println("@AfterAll --- close DB connection ");
    }

    @Before
    public void setUp() {

        System.out.println("@Before Hook---- Launching the browser");
    }

    @After
    public void tearDown() {

        System.out.println("@After Hook---- quitting the browser");
    }
}

```

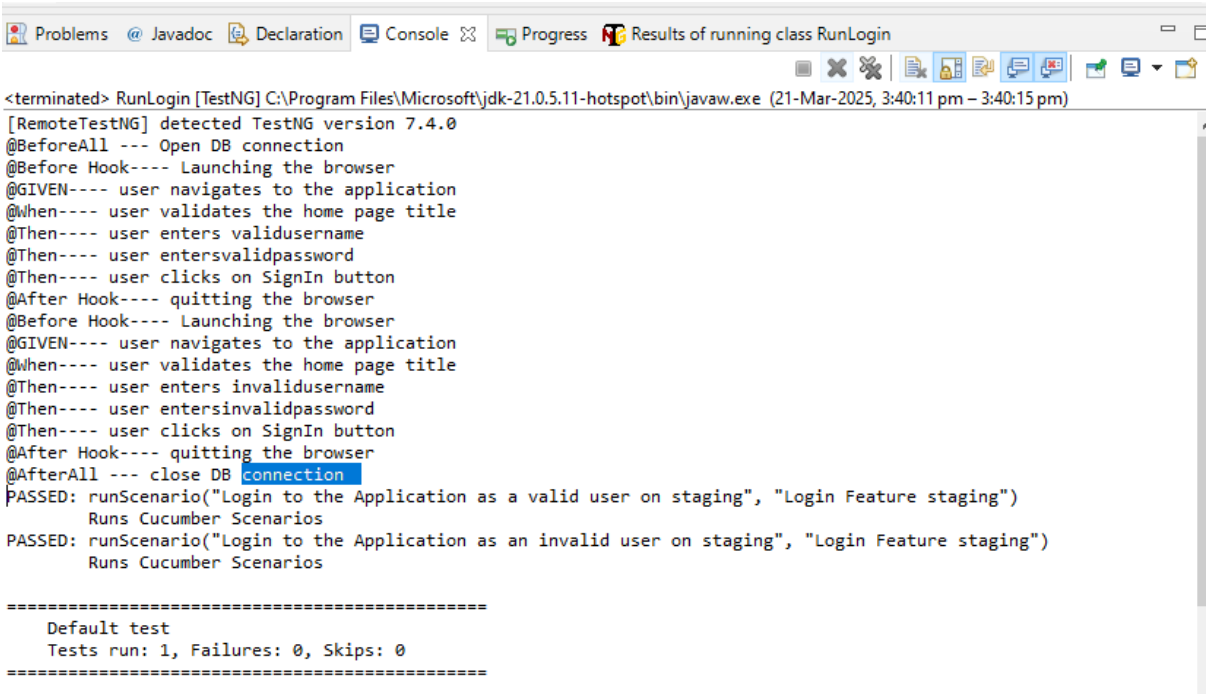
<terminated> RunLogin [TestNG] C:\Program Files\Microsoft\jdk-21.0.5.11-hotspot\bin\javaw.exe (21-Mar-2025, 3:38:29 pm – 3:38:35 pm)

```
[RemoteTestNG] detected TestNG version 7.4.0
FAILED CONFIGURATION: @BeforeClass setUpClass(org.testng.TestRunner@49bf29c6)
io.cucumber.java.InvalidMethodSignatureException: A method annotated with BeforeAll or AfterAll must have one of the:
 * public static void before_or_after_all()
at steps.Hooks.before_All()
```

Make the BeforeAll and AfterAll method as static.

loginBackgroundE... | login.feature | RunLogin.java | testNG.xml | RunLogir

```java
package steps;
import io.cucumber.java.After;
import io.cucumber.java.AfterAll;
import io.cucumber.java.Before;
import io.cucumber.java.BeforeAll;

public class Hooks {

    @BeforeAll
    public static void before_All() {

        System.out.println("@BeforeAll --- Open DB connection ");

    }

    @AfterAll
    public static void after_All() {
        System.out.println("@AfterAll --- close DB connection ");
    }

```



@BeforeStep and @AfterStep – works before and after each steps

```
@BeforeStep
public void before_step() {

    System.out.println("@BeforeStep ---  adding logs");

}

@AfterStep
public void after_step() {
    System.out.println("@BeforeStep ---  deleting logs");

}
```

Problems Javadoc Declaration Console Progress Results of running class RunLogin

<terminated> RunLogin [TestNG] C:\Program Files\Microsoft\jdk-21.0.5.11-hotspot\bin\javaw.exe (21-Mar-2025, 3:47:03 pm – 3:47:07 pm)

```
[RemoteTestNG] detected TestNG version 7.4.0
@BeforeAll --- Open DB connection
@Before Hook---- Launching the browser
@BeforeStep ---  adding logs
@GIVEN---- user navigates to the application
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@When---- user validates the home page title
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@Then---- user enters validusername
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@Then---- user entersvalidpassword
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@Then---- user clicks on SignIn button
@BeforeStep ---  deleting logs
@After Hook---- quitting the browser
@Before Hook---- Launching the browser
@BeforeStep ---  adding logs
@GIVEN---- user navigates to the application
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@When---- user validates the home page title
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@Then---- user enters invalidusername
@BeforeStep ---  deleting logs
@BeforeStep ---  adding logs
@Then---- user entersinvalidpassword
@BeforeStep ---  deleting logs
```

Defining the execution order in Cucumber Hooks

We can have multiple @Before hook and we can define the execution order

User order=0 , 0 being max priority for Before in case of after it will be reverse.

loginBackgroundE... login.feature RunLogin.java testNG.xml RunLoginStaging.ja... Hooks.java

```
10 public class Hooks {
11
12     @BeforeAll
13     public static void before_All() {
14
15         System.out.println("@BeforeAll --- Open DB connection ");
16
17     }
18
19     @AfterAll
20     public static void after_All() {
21         System.out.println("@AfterAll --- close DB connection ");
22     }
23
24     @Before(order = 0)
25     public void setUp1() {
26
27         System.out.println("@Before Hook 1---- Launching the browser");
28     }
29
30     @Before(order = 1)
31     public void setUp2() {
32
33         System.out.println("@Before Hook 2---- Launching the browser");
34     }
35
36     @After(order = 1)
37     public void tearDown1() {
38
39         System.out.println("@After Hook 1---- quitting the browser");
40     }
41
42     @After(order = 0)
43     public void tearDown2() {
```

Tagged Hooks

Before or after condition for a particular tag, make sure runner class is also updated.

```
@Before("@staging")
public void setUp() {

    System.out.println("@Before Hook 1---- Launching the browser");
}
```

```
1  package runner;
2
3  import com.aventstack.extentreports.ExtentReports;
9
10
11 @CucumberOptions(features="src/test/resources/features",
12                 glue= {"steps"},
13                 tags="@production and not @staging",
14                 plugin= {"html:target/cucumber-reports/cucumber-html-report.html",
15                         "com.aventstack.extentreports.cucumber.adapter.ExtentCucumberAdapter:"}
16
17         )
18 public class RunLoginStaging extends AbstractTestNGCucumberTests {
19
20
21
22 }
23
```

Data Table as List

Create a feature file “dataTableAsList.feature

loginBackgroun... | login.feature | DataTableAsList.f... | RunLogin.java | LoginSteps.java | Hooks.java

```
1  @datatable
2  Feature: Login Feature production
3
4    In order to perform successful login
5    As a user
6    I have to enter correct username and password
7
8    @datatable
9    Scenario Outline: Login to the Application with different combinations on production
10     Given user navigates to the application
11     When user validates the home page title
12     Then user enters "valid" username
13     And user enters "valid" password
14     And user validates captcha
15     And user enters firstname and lastname
16       | Rahul | Ritesh |
17       | john  | cena   |
18     And user clicks on SignIn button
19
```

```java
@Then("user enters firstname and lastname")
public void user_enters_firstname_and_lastname(io.cucumber.datatable.DataTable dataTable) {
    // Write code here that turns the phrase above into concrete actions
    // For automatic transformation, change DataTable to one of
    // E, List<E>, List<List<E>>, List<Map<K,V>>, Map<K,V> or
    // Map<K, List<V>>. E,K,V must be a String, Integer, Float,
    // Double, Byte, Short, Long, BigInteger or BigDecimal.
    //
    // For other transformations you can register a DataTableType.
    // data in table is stored in rows and Columns
    List<List<String>> data = dataTable.asLists();
    String firstName = data.get(0).get(0);
    System.out.println("First Name is: "+ firstName);

    String lastName = data.get(0).get(1);
    System.out.println("Last Name is: "+ lastName);

}
```

It is a data table so we can read all the data as row column format.

Data Table As Map

login.feature | DataTableAsL... | RunLogin.java | LoginSteps.java | Hooks.java | DataTableAsM...

```
1  @datatableAsMap
2  Feature: Login Feature production
3
4     In order to perform successful login
5     As a user
6     I have to enter correct username and password
7
8    Scenario Outline: Login to the Application with different combinations on production
9      Given user navigates to the application
10     When user validates the home page title
11     Then user enters "valid" username
12     And user enters "valid" password
13     And user validates captcha
14     And user enters firstname and lastname as map
15       | firstName | lastName |
16       | Rahul     | Ritesh   |
17       | john      | cena     |
18     And user clicks on SignIn button
19
```

```java
@Then("user enters firstname and lastname as map")
public void user_enters_firstname_and_lastname_as_map(io.cucumber.datatable.DataTable dataTable) {
    List<Map<String, String>> map = dataTable.asMaps();
    String firstName = map.get(0).get("firstName");
    String lastName = map.get(0).get("lastName");

    System.out.println("First Name is: "+ firstName + "Last Name is: "+lastName);
    System.out.println("First Name for 2 nd row: "+ map.get(1).get("firstName")
            + "Last Name for 2 nd row: "+ map.get(1).get("lastName"));


}
```

Parameterize data table as a Map

```
@Then("user enters firstname and lastname as map")
public void user_enters_firstname_and_lastname_as_map(io.cucumber.datatable.DataTable dataTable) {
    /*
     * List<Map<String, String>> map = dataTable.asMaps(); String firstName =
     * map.get(0).get("firstName"); String lastName = map.get(0).get("lastName");
     *
     * System.out.println("First Name is: "+ firstName + "Last Name is: "+lastName);
     * System.out.println("First Name for 2 nd row: "+ map.get(1).get("firstName") +
     * "Last Name for 2 nd row: "+ map.get(1).get("lastName"));
     */

    for(Map<String,String> data : dataTable.asMaps(String.class,String.class)) {

        System.out.println("FirstName is----"+data.get("firstName")
        + "  LastName is----"+data.get("lastName"));


    }


}
```

Taking Screenshot on failure

```
package steps;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;

import io.cucumber.java.Scenario;
import io.cucumber.java.After;
import utils.SeleniumDriver;

public class AfterActions {

    @After
    public static void tearDown( @NotNull Scenario scenario) {

        WebDriver driver=SeleniumDriver.getDriver();
        System.out.println(scenario.isFailed());
         if (scenario.isFailed()) {
             byte[] screenshotBytes = ((TakesScreenshot) driver).getScreenshotAs( target: OutputType.BYTES);
             scenario.attach( data: screenshotBytes,  mediaType: "image/png", name: "image");

         }
    SeleniumDriver.tearDown();
    }
}
```

```java
package steps;

import io.cucumber.java.Before;
import utils.SeleniumDriver;

public class BeforeActions  {

    @Before
    public static void setUp() {
        //ExtentCucumberFormatter.initiateExtentCucumberFormatter();
        System.out.println("Before");
       SeleniumDriver.setUpDriver();
    }
}
```