Maven Cucumber Test Framework

Install base softwares

After you have administrative rights install the following software (on Windows):

- Java 8 JDK - (https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html)
- Java 11 JDK - (Archived OpenJDK GA Releases)

For installation instructions please visit this doc : Install Open JDK 11

- Maven (~3.8.1) - (http://maven.apache.org/install.html)
- Download intellij IDEA (https://www.jetbrains.com/idea/download)

After Installing

Upon finalizing the install of java, maven, and docker you will want to add the path of these files to your environment variables on your computer.

There is a method for install on Windows and macOS:

Windows Variable Installation

1. Open the Command Prompt as *Administrator*
2. Set the variable name and value by executing the following command:

1setx /M MY_VARIABLE_NAME "my_variable_value"

1. Restart the Command Prompt to reload the environment variables and check the new variable as follows:

1echo %MY_VARIABLE_NAME%

Note: Once set, calling echo should return the variable location.

macOS Variable Installation

The variables need to be added in the .bash_profile file or .zshrc file depending on the Terminal type you are using, for example:

1export JAVA_HOME="$(/usr/libexec/java_home -v1.8)"

2export M2_HOME=/opt/apache-maven-3.8.1

3export M2=$M2_HOME/bin

4export PATH=$PATH:$M2

Reopen your Terminal to see changes take effect (or run source ~/.bash_profile or source ~/.zshrc)

Once all the steps have concluded, you can check that everything is installed correctly by running:

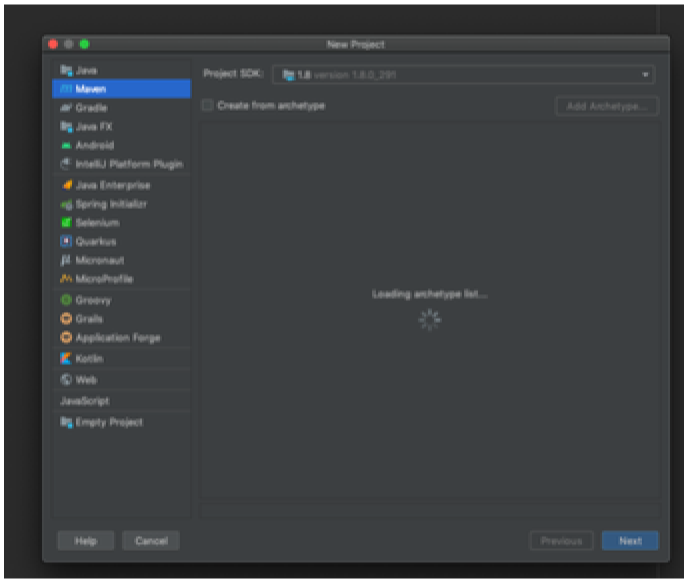
1-javac -version or java -v -> **Java v8 or later**

2-mvn –version -> **Apache Maven 3.8.1**

-----------------------/---------------------

STEPS:

1- Crete new maven project.



Project SDK: 1.8 version 1.8.0_291

Create from archetype

2- add all your test framework dependencies in pom.xml file that will be added after you generate your fist Maven project.

- Cucumber-java:

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-java -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>6.10.4</version>
</dependency>
```

- Maven

```
<!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-
dependency-plugin -->
<dependency>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifactId>
    <version>3.1.1</version>
</dependency>
```

- Cucumber-Junit

```
<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->
<dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>6.9.0</version>
    <scope>test</scope>
</dependency>
```

- Junit

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
</dependency>
```

- Selenium

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium
/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

- WebDriver Manager

```
<!-- https://mvnrepository.com/artifact/io.github.bonigarcia
/webdrivermanager -->
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>4.4.3</version>
</dependency>
```

- Execute Test from command line

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.19.1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
        <properties>
            <property>
                <name>junit</name>
                <value>true</value>
            </property>
        </properties>
        <includes>
            <include>**/*Runner.java</include>
        </includes>
    </configuration>
</plugin>
```

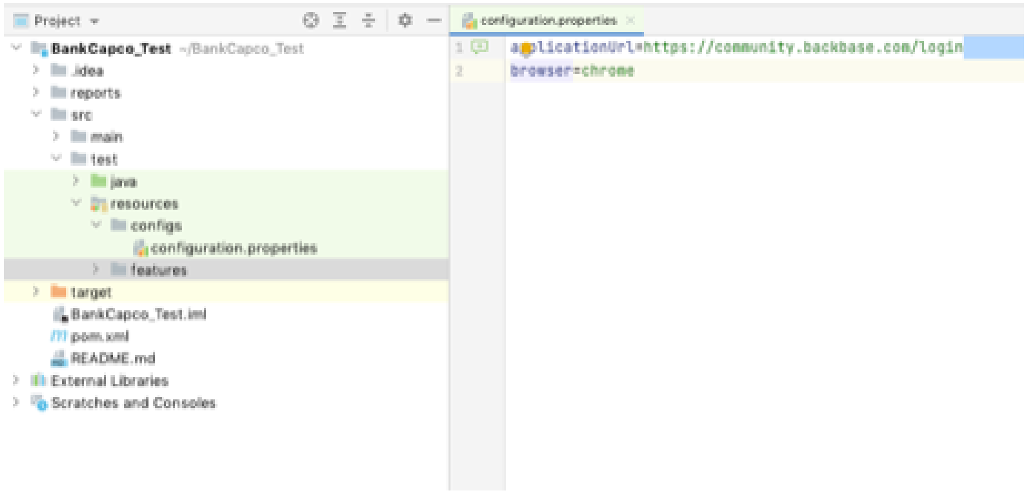
- Cucumber Reporting

```
<plugin>
    <groupId>net.masterthought</groupId>
    <artifactId>maven-cucumber-reporting</artifactId>
    <version>5.5.4</version>
    <executions>
        <execution>
            <id>execution</id>
            <phase>verify</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <projectName>BankCapco_Test</projectName>
                <!-- output directory for the generated report -->
                <outputDirectory>${project.build.directory}<
/outputDirectory>
                <!-- optional, defaults to outputDirectory if not
specified -->
                <inputDirectory>${project.build.directory}<
/inputDirectory>
                <jsonFiles>
                    <!-- supports wildcard or name pattern -->
                    <param>**/*.json</param>
                </jsonFiles>
            </configuration>
        </execution>
    </executions>
</plugin>
```

3- Add resources folder .

4- Create config and features folder under resources folder.

5- Create new properties file in config folder. ex: configuration.properties



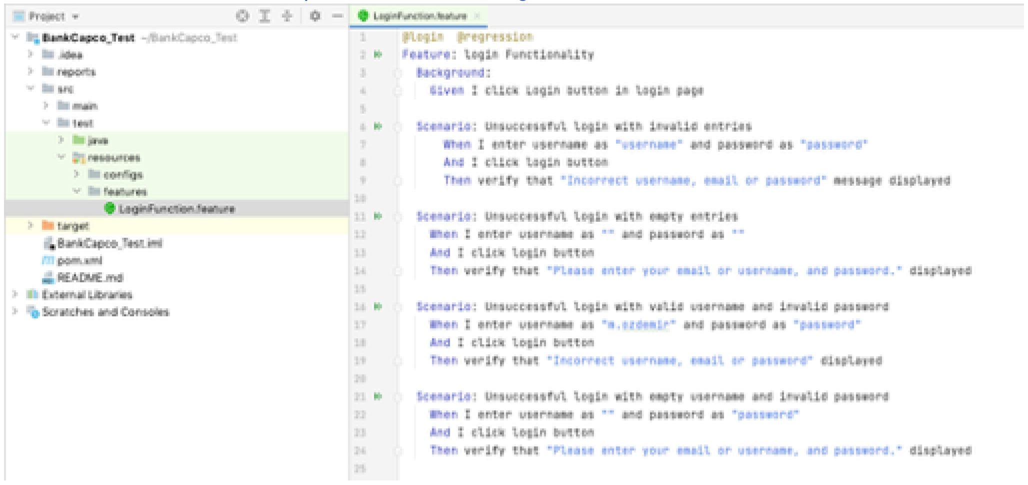
```
applicationUrl=https://community.backbase.com/login
browser=chrome
```

```
applicationUrl=https://community.backbase.com/login
browser=chrome
```

6- Create feature file in features folder. ex: LoginFucnctions.feature

A ***Feature File*** is an entry point to the Cucumber tests. This is a file where you will describe your tests in Descriptive language (*Like English*). It is an essential part of Cucumber, as it serves as an automation test script as well as live documents. A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios.

For more info about Gherkin: https://cucumber.io/docs/gherkin/reference/



```
@login  @regression
Feature: login Functionality
  Background:
    Given I click login button in login page

  Scenario: Unsuccessful login with invalid entries
      When I enter username as "username" and password as "password"
      And I click login button
      Then verify that "Incorrect username, email or password" message displayed

  Scenario: Unsuccessful login with empty entries
    When I enter username as "" and password as ""
    And I click login button
    Then verify that "Please enter your email or username, and password." displayed

  Scenario: Unsuccessful login with valid username and invalid password
    When I enter username as "m.ozdemir" and password as "password"
    And I click login button
    Then verify that "Incorrect username, email or password" displayed

  Scenario: Unsuccessful login with empty username and invalid password
    When I enter username as "" and password as "password"
    And I click login button
    Then verify that "Please enter your email or username, and password." displayed
```

```
@login  @regression
Feature: login Functionality

  Background:
    Given I click Login button in login page

  Scenario: Unsuccessful login with invalid entries
    When I enter username as "username" and password as "password"
    And I click login button
    Then verify that "Incorrect username, email or password" message
displayed

  Scenario: Unsuccessful login with empty entries
    When I enter username as "" and password as ""
    And I click login button
    Then verify that "Please enter your email or username, and
password." displayed

  Scenario: Unsuccessful login with valid username and invalid password
    When I enter username as "m.ozdemir" and password as "password"
    And I click login button
    Then verify that "Incorrect username, email or password" displayed

  Scenario: Unsuccessful login with empty username and invalid password
    When I enter username as "" and password as "password"
    And I click login button
    And I click next button
    Then verify that "Please enter your email or username, and
password." displayed

  @test
  Scenario:Successful login with valid credentials
    Given I enter user name as "m.ozdemir" and password as "Capco@3033"
    And I click login button
    Then I verify greeting message "Hi musa ozdemir" is displayed in
dashboard
```

7- Create following Packages in Java folder. 1- pages 2- runners 3- stepDefinitions 4- testBase 5-utils

Pages:

We are using **Page Object Model (POM) and Page Factory.**

Page Object Model (POM) is a design pattern, popularly used in test automation that creates Object Repository for web UI elements. The advantage of the model is that it reduces code duplication and improves test maintenance. Under this model, for each web page in the application, there should be a corresponding Page Class.

Page Factory in Selenium is an inbuilt Page Object Model framework concept for Selenium WebDriver but it is very optimized. It is used for initialization of Page objects or to instantiate the Page object itself. It is also used to initialize Page class elements without using “FindElement/s.”

Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of class PageFactory in Selenium, we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

@FindBy can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath** as attributes.

1. Create a ***'New Package'*** file and name it as "***pages***", by right click on the Project and select ***New > Package.***

2. Create a ***'New Class'*** file, by right click on the above-created Package and select ***New > Class*** and name it as ***xxxxxPageElements.***

3. Store webElements inside the class using with page factory.

Ex:

LoginPageElements:

```
package pages;

import org.openqa.selenium.By;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import testBase.PageInitializer;

public class LoginPageElements extends PageInitializer {

    @FindBy(id = "login-account-name")
    public WebElement userNameTextBox;

    @FindBy(id = "login-account-password")
    public WebElement passwordTextBox;

    @FindBy(xpath = "//*[@class=\"btn btn-large btn-primary btn btn-
icon-text ember-view\"]")
    public WebElement loginButton;

    @FindBy(xpath = "//*[@id=\"modal-alert\"]")
    public WebElement allertMessage;

public LoginPageElements(){
    PageFactory.initElements(driver , this);
}
}
```

Utils:

Constants:

1. Create a ***'New Package'*** file and name it as "***utility***", by right click on the Project and select ***New > Package.***
2. Create a ***'New Class'*** file, by right click on the above-created Package and select ***New > Class*** and name it as ***Constant.***
3. Assign keywords in this class to your fixed data for e.g. filePaths and wait times.

```
package utils;

public class Constants {

    public static final String CONFIGURATION_FILEPATH=System.getProperty
("user.dir")+"/src/test/resources/configs/configuration.properties";
    public static final int IMPLICIT_WAIT_TIME = 20;
    public static final int EXPLICIT_WAIT_TIME = 20;
}
```

Constants Variables are declared as ***public static***, so that they can be called in any other methods without ***instantiate*** the class.

Constant Variables are declared a ***final***, so that they cannot be changed during the execution.

ConfigsReader:

This class helps us to read configuration files.

1. Create a ***New Class*** file and name it as ***ConfigsReader***, by *right click* on the above-created package and select ***New >> Class***.

```
package utils;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Properties;

public class ConfigsReader {

    static Properties prop;

    public static void readProperties(String filePath){
        try{
            FileInputStream fis=new FileInputStream(filePath);
            prop=new Properties();
            prop.load(fis);
            fis.close();

        } catch (FileNotFoundException e) {
            e.printStackTrace();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static String getPropValue(String key){
        return prop.getProperty(key);
    }
}
```

Common Methods:

Create common methods Java class and extends to PageInitilazer class. Store common methods in this class. Make methods **static** for easy to call by class name.

1. Create a ***New Class*** file and name it as ***CommonMethods,*** by *right click* on the above-created package and select ***New >> Class***.

```
package utils;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
import testBase.PageInitializer;

public class CommonMethods extends PageInitializer {
    /**
     * Method that send text to any given elements
     * @param element
     * @param text
     */
    public static void sendText(WebElement element, String text){
        element.clear();
        element.sendKeys(text);
    }

    public static void click(WebElement element){
        waitForClickability(element);
        element.click();
    }
    public static WebDriverWait getWaitObject(){
        return new WebDriverWait(driver , Constants.EXPLICIT_WAIT_TIME);
    }

    public static void waitForClickability(WebElement element){
        getWaitObject().until(ExpectedConditions.elementToBeClickable
(element));
    }
}
```

Test Base:

BaseClass:

1. Create a ***New Class*** file and name it as ***BaseClass***, by *right click* on the above-created package and select ***New >> Class***.
2. Create setup and teardown methods.

```
package testBase;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import utils.Constants;
import utils.ConfigsReader;
import java.util.Locale;
import java.util.concurrent.TimeUnit;

public class BaseClass {
    protected static WebDriver driver;

    public static void setUp() {
        ConfigsReader.readProperties(Constants.CONFIGURATION_FILEPATH);

        switch (ConfigsReader.getPropValue("browser").toLowerCase
(Locale.ROOT)) {

            case "chrome":
                WebDriverManager.chromedriver().setup();
                driver = new ChromeDriver();
                break;

            case "firefox":
                WebDriverManager.firefoxdriver().setup();
                driver = new FirefoxDriver();
                break;

            case "edge":
                WebDriverManager.edgedriver().setup();
                driver = new EdgeDriver();
                break;

            default:

                throw new RuntimeException("Browser is not supported");
        }

        driver.manage().timeouts().implicitlyWait(Constants.
IMPLICIT_WAIT_TIME, TimeUnit.SECONDS);
        driver.get(ConfigsReader.getPropValue("applicationUrl"));
        PageInitializer.initializePageObjects();
    }

    public static void tearDown(){

        if(driver != null){
            driver.quit();
        }
    }
}
```

Page Initializer:

1. Create a ***New Class*** file and name it as ***PageInitializer***, by *right click* on the above-created package and select ***New >> Class***.
2. Extends PageInitializer class to BaseClass.
3. Create class object for each class in the Pages packages.
4. Create **initializePageObjects** method.

```
package testBase;

import pages.DashboardPageElements;
import pages.LoginPageElements;
import pages.MainLoginPageElements;

public class PageInitializer extends BaseClass {

    public static LoginPageElements login;
    public static DashboardPageElements dash;
    public static MainLoginPageElements main;

    public static void initializePageObjects(){
        login = new LoginPageElements();
        dash = new DashboardPageElements();
        main = new MainLoginPageElements();
    }
}
```

Runners:

RegressionRunner:

1. Create a ***New Class*** file and name it as ***RegressionRunner***, by *right click* on the above-created package and select ***New >> Class***.

```
package runners;

import org.junit.runner.RunWith;
import io.cucumber.junit.CucumberOptions;
import io.cucumber.junit.Cucumber;

@RunWith(Cucumber.class)
@CucumberOptions(
        features = "src/test/resources/features" , // ->set: path to
the feature files
        glue = "stepDefinitions", // -> set: path to the step
definitions
        dryRun = false,// -> true: checks if all the steps have step
definition
        monochrome = true, // -> true: display console output is more
readable fromat
        tags = "@regression", // -> what tags in feature files should
be executed
        plugin = {"pretty", // -> set: what all report format to use
           "html:target/cucumber-default-reports", // -> generate
default html reports
           "rerun:target/failedTest.txt",// ->generates txt file only
with failed test
           "json:target/cucumber.json" // -> generates json reports
        }
)

public class RegressionRunner {
}
```

2. Set dryRun **false** to check all steps have step in step definitions. If there is any copy from console undefined steps and paste to related step definitions file.

Step Definitions:

Hooks:

1. Create a ***New Class*** file and name it as ***Hooks***, by *right click* on the above-created package and select ***New >> Class***.
2. Create startTest and endTest methods. And use @Before tags to run methods before each scenario and @After tags to run methods after each scenario.

```
package stepDefinitions;

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

public class Hooks {

    @Before
    public void startTest() {
        BaseClass.setUp();
    }

    @After
    public void endTest() {
        BaseClass.tearDown();
    }
}
```

LoginFunctionStepDefinitions:

1. Create a ***New Class*** file and name it as ***LoginFunctionStepsDefinitions***, by *right click* on the above-created package and select ***New >> Class***. And Extends class to **CommonMethods**.
2. Write actual code inside to each methods to execute the test scenario in the Features file.

```
package stepDefinitions;

import io.cucumber.gherkin.internal.com.eclipsesource.json.PrettyPrint;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;
import org.junit.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.support.ui.ExpectedConditions;
import pages.MainLoginPageElements;
import utils.CommonMethods;
import utils.ConfigsReader;

public class LoginFunctionsStepDefinitions extends CommonMethods {

    @Given("I click Login button in login page")
    public void i_click_login_button_in_login_page() {

        click(main.mainPageLoginButton);
    }

    @When("I enter username as {string} and password as {string}")
    public void i_enter_username_as_and_password_as(String username,
String password) {

        sendText(login.userNameTextBox, username);
        sendText(login.passwordTextBox, password);
    }

    @When("I click login button")
    public void i_click_login_button() {

        click(login.loginButton);
    }

    @Then("verify that {string} message displayed")
    public void verify_that_message_displayed(String allertMessage)
throws InterruptedException {

       Thread.sleep(2000);
       Assert.assertEquals(allertMessage, login.allertMessage.
getText());

    }

    @Then("verify that {string} displayed")
    public void verify_that_displayed(String allertMessage) throws
InterruptedException {

        Thread.sleep(2000);
        Assert.assertEquals(allertMessage, login.allertMessage.
getText());
    }

    @Given("I enter user name as {string} and password as {string}")
    public void i_enter_user_name_as_and_password_as(String username,
String password) {

        sendText(login.userNameTextBox, username);
        sendText(login.passwordTextBox, password);
    }

    @Then("I verify greeting message {string} is displayed in
dashboard")
    public void i_verify_greeting_message_is_displayed_in_dashboard
(String message) throws InterruptedException {

        Thread.sleep(2000);
        Assert.assertEquals(message, dash.greetingMessage.getText());
    }
}
```

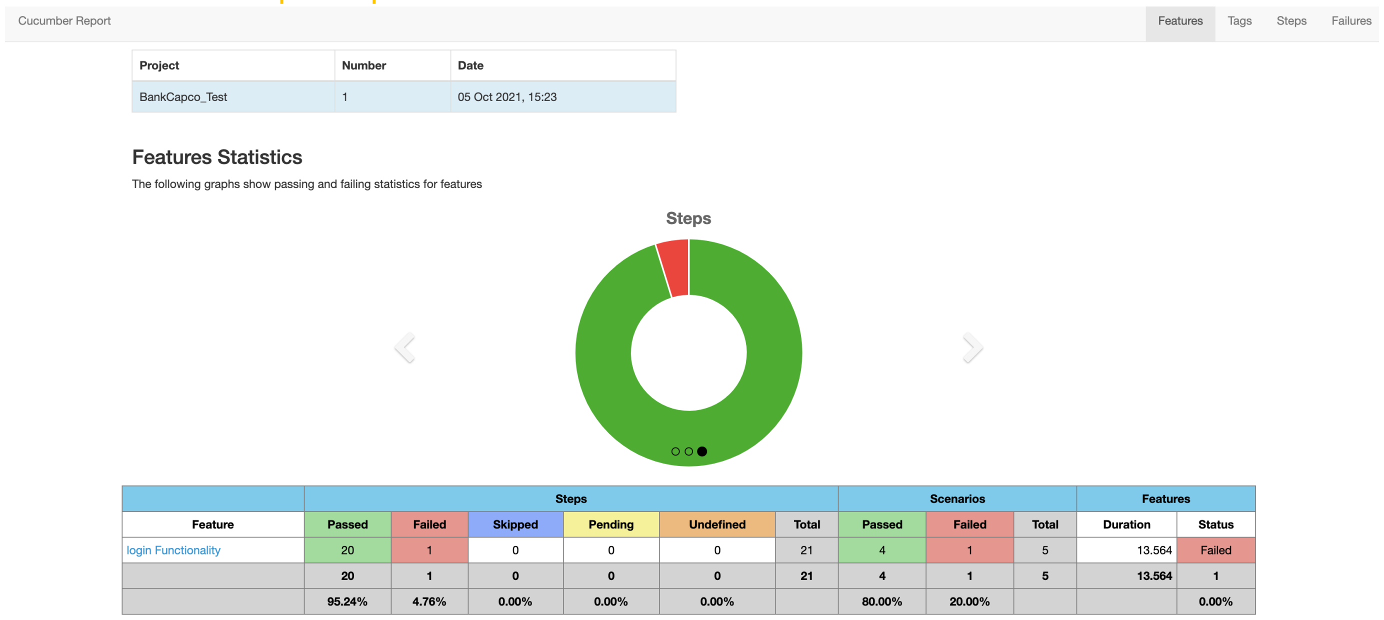
Running Scenarios using Tags from Command Line

If you are using Maven and want to run a subset of scenarios tagged with @test.

```
mvn test -Dcucumber.filter.tags="@test"
```

[illegible]

Maven Cucumber HTML Report Sample:



Cucumber Report — Features | Tags | Steps | Failures

Project	Number	Date
BankCapco_Test	1	05 Oct 2021, 15:23

Features Statistics

The following graphs show passing and failing statistics for features

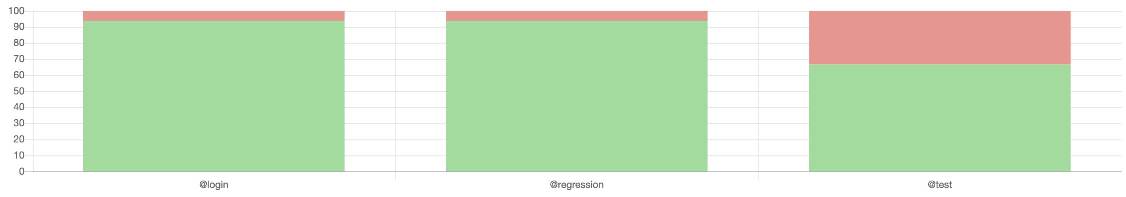
	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
login Functionality	20	1	0	0	0	21	4	1	5	13.564	Failed
	20	**1**	**0**	**0**	**0**	**21**	**4**	**1**	**5**	**13.564**	**1**
	95.24%	**4.76%**	**0.00%**	**0.00%**	**0.00%**		**80.00%**	**20.00%**			**0.00%**

Cucumber Report — Features | Tags | Steps | Failures

Project	Number	Date
BankCapco_Test	1	05 Oct 2021, 15:23

Tags Statistics

The following graph shows passing and failing statistics for tags



	Steps						Scenarios			Features	
Tag	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
@login	15	1	0	0	0	16	4	1	5	12.691	Failed
@regression	15	1	0	0	0	16	4	1	5	12.691	Failed
@test	2	1	0	0	0	3	0	1	1	3.368	Failed
	32	3	0	0	0	35	0	11	11	28.752	3
	91.43%	8.57%	0.00%	0.00%	0.00%		0.00%	100.00%			0.00%

Cucumber Report — Features | Tags | Steps | Failures

Project	Number	Date
BankCapco_Test	1	05 Oct 2021, 15:23

Steps Statistics

The following graph shows step statistics for this build. Below list is based on results. step does not provide information about result then is not listed below. Additionally @Before and @After are not counted because they are part of the scenarios, not steps.

Implementation	Occurrences	Average duration	Max duration	Total durations	Ratio
`stepDefinitions.Hooks.endTest()`	5	0.142	0.175	0.714	100.00%
`stepDefinitions.Hooks.startTest()`	5	3.983	4.998	19.917	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_click_login_button()`	5	0.106	0.133	0.531	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_click_login_button_in_login_page()`	5	0.174	0.238	0.873	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_click_next_button()`	1	0.000	0.000	0.000	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_enter_user_name_as_and_password_as(java.lang.String,java.lang.String)`	1	0.221	0.221	0.221	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_enter_username_as_and_password_as(java.lang.String,java.lang.String)`	4	0.199	0.240	0.797	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.i_verify_greeting_message_is_displayed_in_dashboard(java.lang.String)`	1	3.044	3.044	3.044	0.00%
`stepDefinitions.LoginFunctionsStepDefinitions.verify_that_displayed(java.lang.String)`	3	2.023	2.024	6.070	100.00%
`stepDefinitions.LoginFunctionsStepDefinitions.verify_that_message_displayed(java.lang.String)`	1	2.026	2.026	2.026	100.00%
10	31	1.103	19.917	34.196	Totals

Cucumber Report — Features | Tags | Steps | Failures

Project	Number	Date
BankCapco_Test	1	05 Oct 2021, 15:23

Failures Overview

The following summary displays scenarios that failed.

Feature: login Functionality
Tags: @login @regression @test
Scenario Successful login with valid credentials — 3.368
Hooks
Before `stepDefinitions.Hooks.startTest()` — 3.762
Steps
Given I enter user name as **"m.ozdemir"** and password as **"Capco@3033"** — 0.221
And I click login button — 0.102
Then I verify greeting message **"Hi musa ozdemirrrr"** is displayed in dashboard — 3.044

org.junit.ComparisonFailure:

```
org.junit.ComparisonFailure: expected:<Hi musa ozdemir[rrr]> but was:<Hi musa ozdemir[]>
	at org.junit.Assert.assertEquals(Assert.java:117)
	at org.junit.Assert.assertEquals(Assert.java:146)
	at stepDefinitions.LoginFunctionsStepDefinitions.i_verify_greeting_message_is_displayed_in_dashboard(LoginFunctionsStepDefinitions.java:64)
	at ✱.I verify greeting message "Hi musa ozdemirrrr" is displayed in dashboard(file:///Users/mozm/BankCapco_Test/src/test/resources/features/LoginFunction.
```

Hooks
After `stepDefinitions.Hooks.endTest()` — 0.174