Automation Testing for UI Application

Agitha Pramesti Sembiring / Associate SDET at Blibli.com

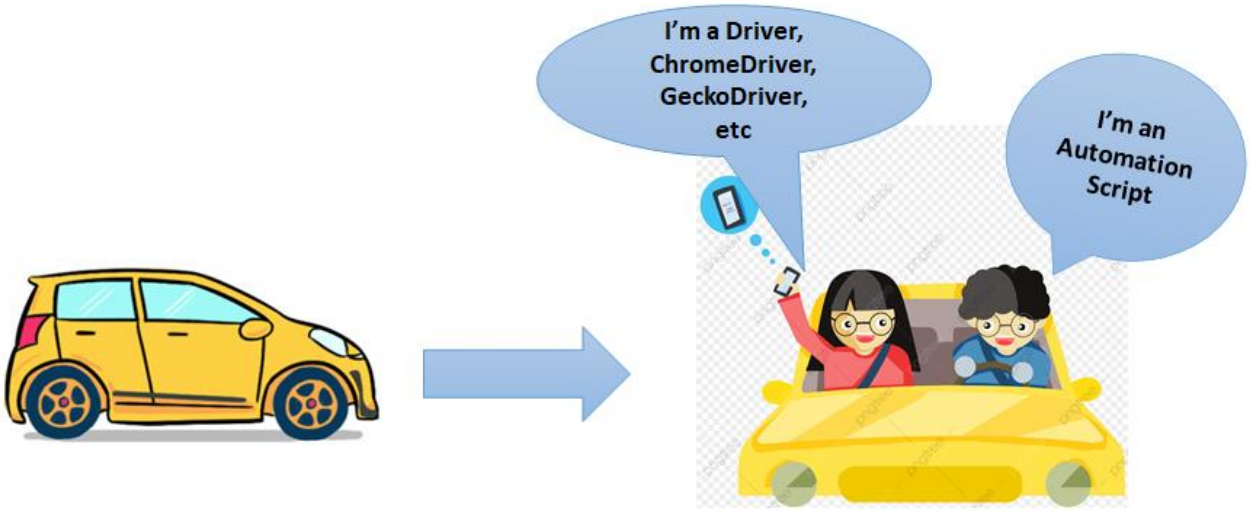
Introduction

Automation pada dasarnya merupakan penciptaan dan pengaplikasian dari teknologi untuk memproduksi dan men-*deliver service-service* dengan intervensi manusia seminimal mungkin.

Ketika suatu aplikasi di*develop*, beberapa tim disatukan untuk membuat aplikasi tersebut stabil. QA/Testers memulai dengan memeriksa dokumen yang berisi *requirements* atau kebutuhan-kebutuhan serta fungsi yang akan di*develop* pada aplikasi tersebut dan membuat *test cases* dari dokumen tersebut. Di Blibli.com, kita menggunakan beberapa *tools* untuk melakukan automation ui yaitu Selenium, BDD Testing, Cucumber, dan Serenity BDD.

- **Selenium**

Selenium merupakan *testing tools* yang berbasis JavaScript Framework, dapat menjalankan *scenario/testcase* secara langsung ke *browser*(Google, Firefox, dll) yang kita inginkan.



- **BDD Testing**

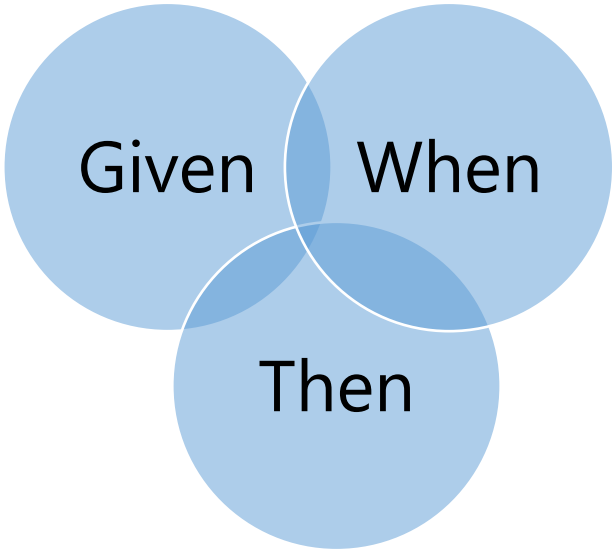
Behavior Driven Development merupakan *software test* yang adalah pengembangan dari TDD, menggunakan *human-readable descriptions*, atau yang biasa dikenal sebagai *Gherkin*. BDD Testing memiliki format yaitu 'Given-When-Then', sebagai contoh:

Given a certain scenario
When an action takes place
Then this should be the outcome.

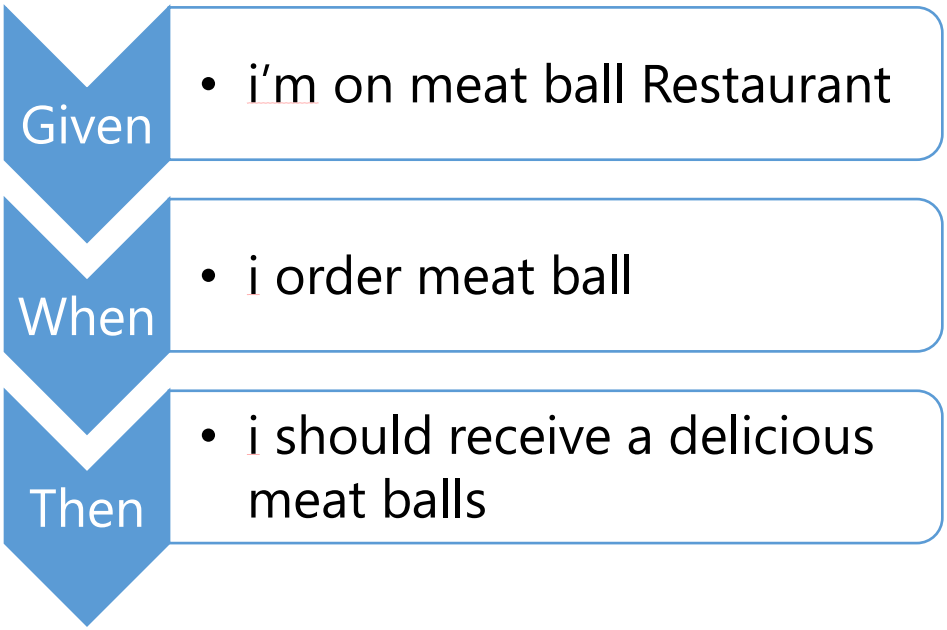
https://medium.com/javascript-scene/behavior-driven-development-bdd-and-functional-testing-62084ad7f1f2

Scenario BDD

1. *Keyword* Given digunakan untuk Initial context (*Open Blibli Page*, dll)
2. *Keyword* When digunakan untuk eventnya (*Click button login*, dll)
3. *Keyword* Then digunakan untuk *assertion*



Contoh Penulisan Scenario / Testcase



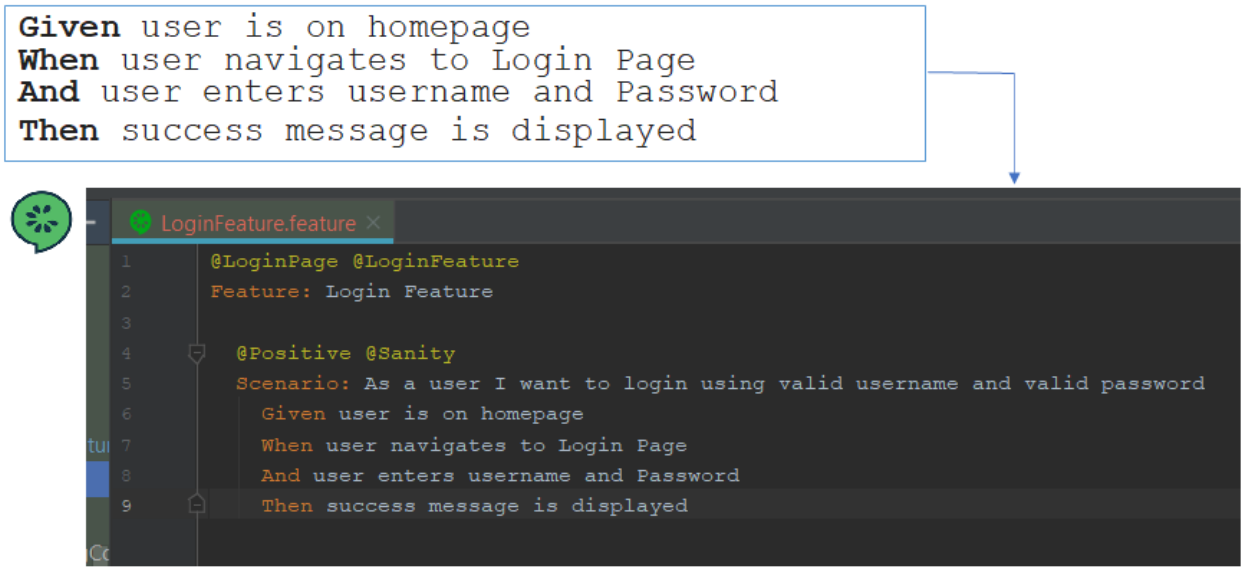
Initial context on Meat ball R estaurant

Do Action Order meat ball

Assertion For Receive deli cious Meat balls

- **Cucumber**

Cucumber merupakan *open source tool* yang mendukung Behavior Driven Development (BDD) *framework*. Bagaimana BDD bekerja pada Cucumber Automation?



https://www.guru99.com/introduction-to-cucumber.html

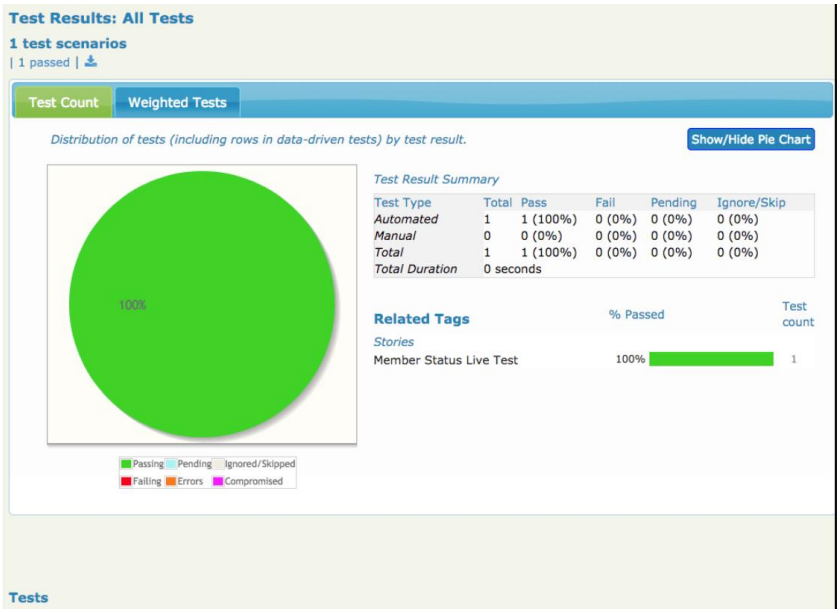
- **Serenity BDD**

Serenity BDD merupakan *tools* yang sangat berguna untuk membuat *automation test web*, *mobile*, dan *api test*.

Serenity BDD merupakan *open souce library* yang membantu dalam mengenerate *well-illustrated testing report*. Serenity tidak hanya menampilkan hasil reports test dari automation kita, namun juga dapat membuat dokumentasi secara berkala.

https://serenity-bdd.github.io/theserenitybook/latest/index.html

https://www.baeldung.com/serenity-bdd



Contoh hasil report Serenity

How automation works ?

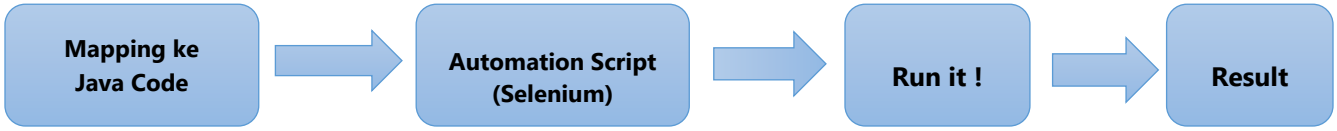
Story: I want To eat delicious meat balls

Scenario 1: Go to Restaurant and eat meat balls

Given I'm on Meat ball Restaurant

When I order meat ball

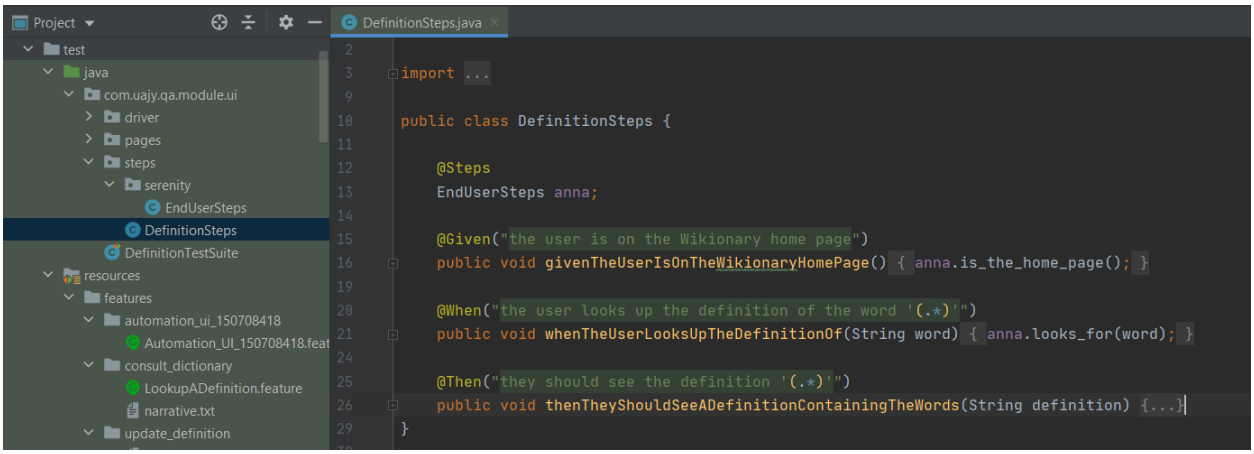
Then I should receive a delicious meat balls



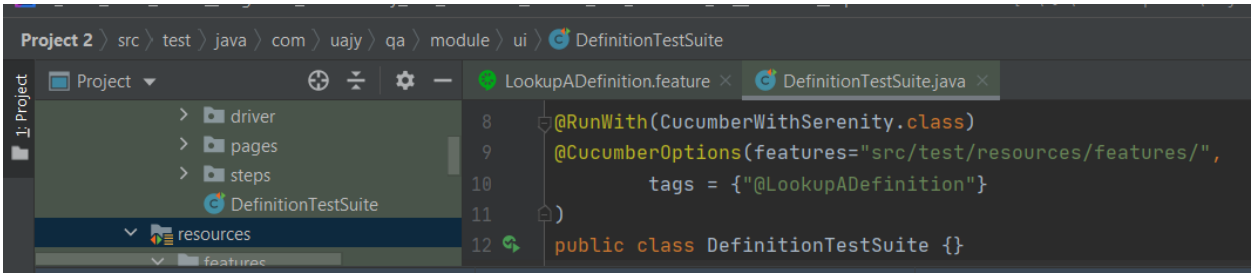
Implementasi Dalam Project :

```
@LookupADefinition
Feature: Lookup a definition
  In order to talk better
  As an English student
  I want to look up word definitions

  Scenario: Looking up the definition of 'apple'
    Given the user is on the Wikionary home page
    When the user looks up the definition of the word 'apple'
    Then they should see the definition 'A common, round fruit produced by the tree Malus domestica, cultivated in temperate climates.'

  Scenario: Looking up the definition of 'pear'
    Given the user is on the Wikionary home page
    When the user looks up the definition of the word 'pear'
    Then they should see the definition 'An edible fruit produced by the pear tree, similar to an apple but elongated towards the stem.'
```

Scenario/Test Case dituliskan secara dengan Cucumber BDD dalam file dengan *extension* '.feature'



Di*mappingkan* ke dalam bahasa Java melalui kelas Steps

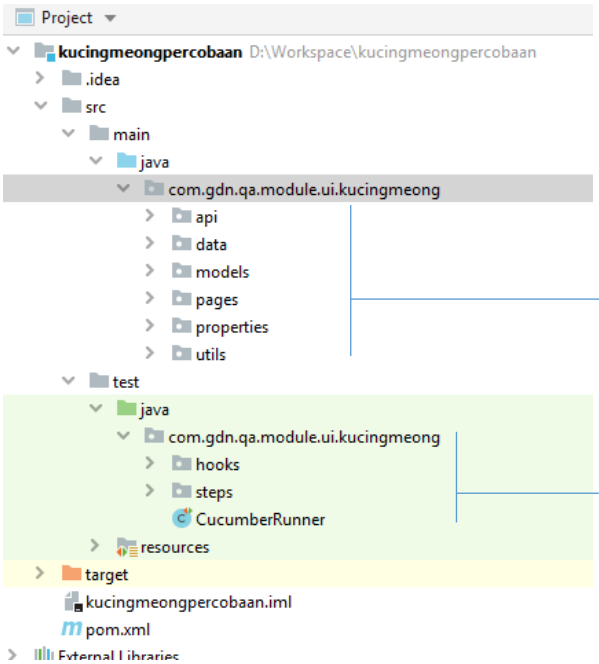


Lalu Jalankan dengan memanggil nama featurenya

Steps	Screenshots	Outcome	
Given the user is on the Wikionary home page		SUCCESS	10.52s
When the user looks up the definition of the word 'apple'		SUCCESS	5.15s
Then they should see the definition 'A common, round fruit produced by the tree Malus domestica, cultivated in temperate climates.'		SUCCESS	0.84s
		SUCCESS	*16.71s*

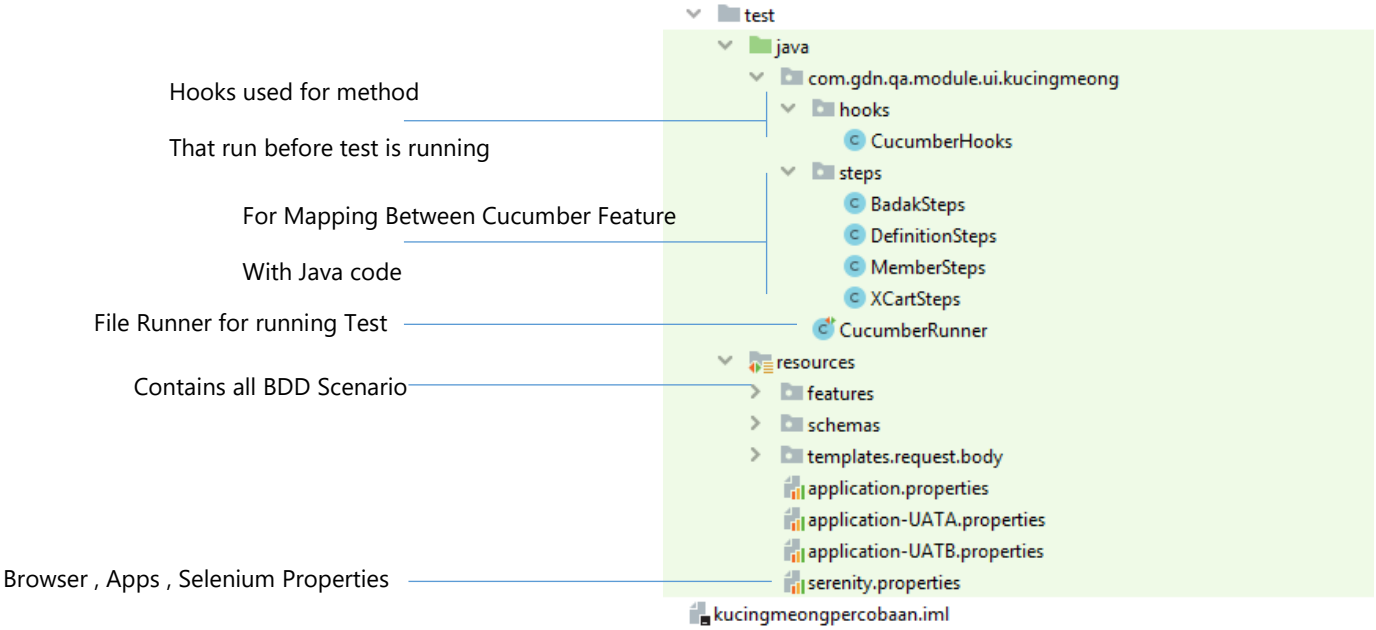
Berikut hasil report automation ui.

Struktur Folder Automation



Main Package, For Action (Hit API) & UI Intercation etc.

Test Package, for logic automation, Mapping Between Feature files with Java code

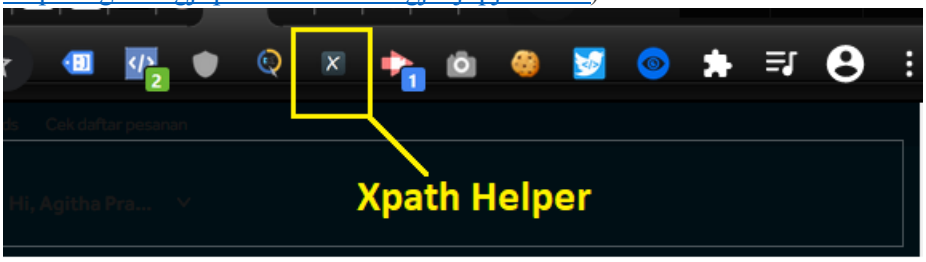

Hooks used for method
That run before test is running
For Mapping Between Cucumber Feature
With Java code
File Runner for running Test
Contains all BDD Scenario
Browser , Apps , Selenium Properties
test
java
com.gdn.qa.module.ui.kucingmeong
hooks
CucumberHooks
steps
BadakSteps
DefinitionSteps
MemberSteps
XCartSteps
CucumberRunner
resources
features
schemas
templates.request.body
application.properties
application-UATA.properties
application-UATB.properties
serenity.properties
kucingmeongpercobaan.iml

DEMO 😊

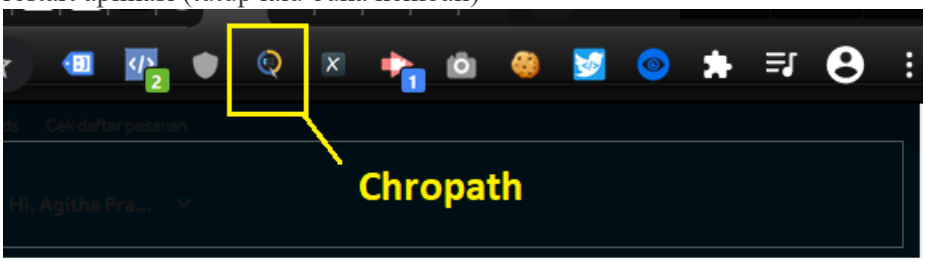
Langkah-langkah

Sebelum memulai pastikan memiliki :

- IDE dan telah melakukan setup/install keperluan seperti ppt panduan yang telah diberikan
- Tambahkan extension "**Xpath Helper**" pada google browser (https://chrome.google.com/webstore/detail/xpath-helper/hgimnogjllphhhkhlmebbmlgjoejdpjl?hl=en)

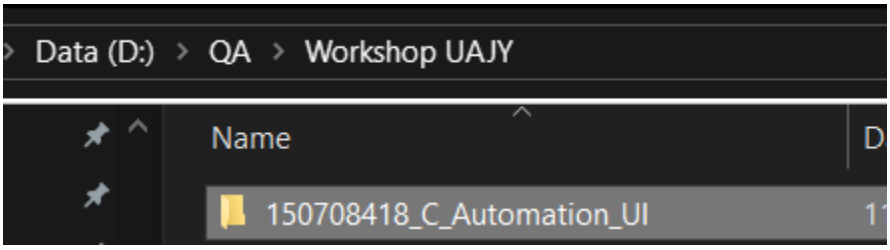


- Tambahkan extension "**Cropath**" pada google browser (https://chrome.google.com/webstore/detail/chropath/ljngjbnaijcbncmcnjfhigebomdlkcjo/) lalu restart aplikasi (tutup lalu buka kembali)



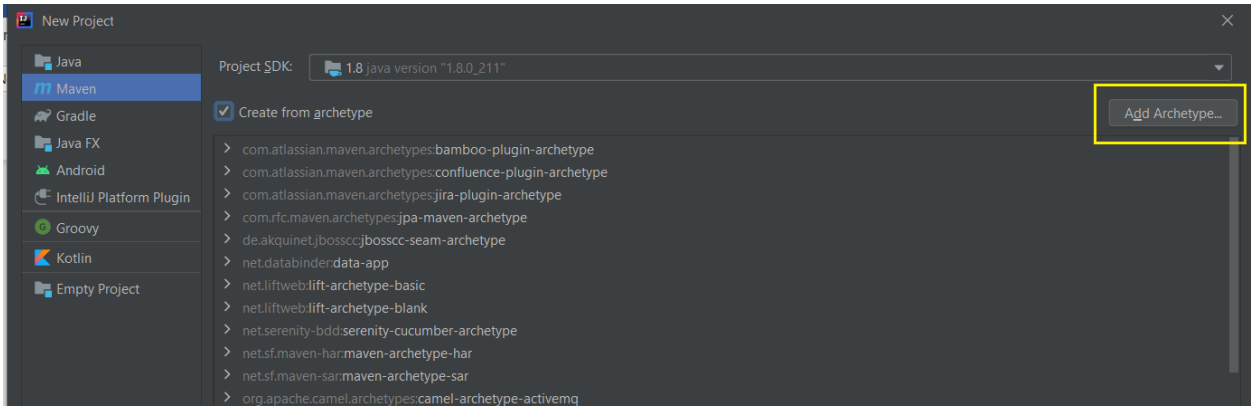
Format Penamaan : NPM_Kelas_Automation_UI

1. Buat folder bernama '150708418_C_Automation_UI'



2. Buka IntelJ dan klik New Project/New Project

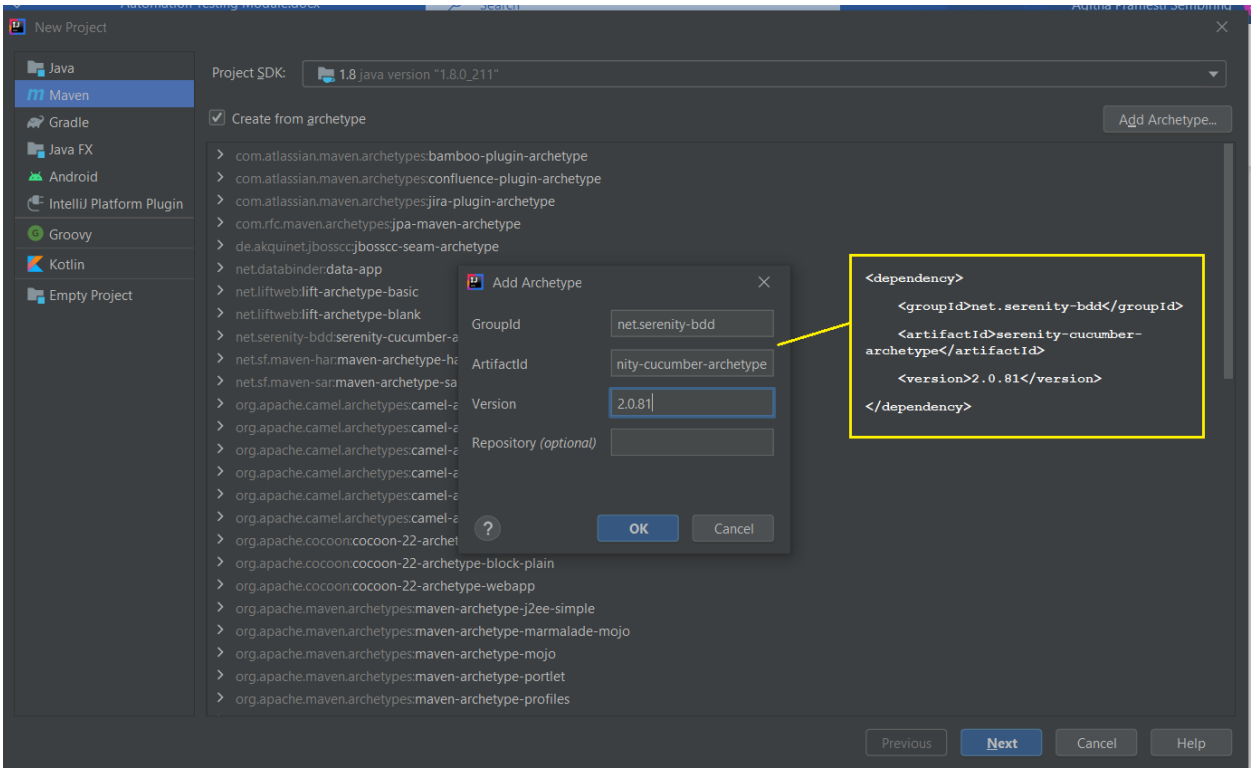
3. Tambahkan Archetype dengan klik '**Add Archetype**'



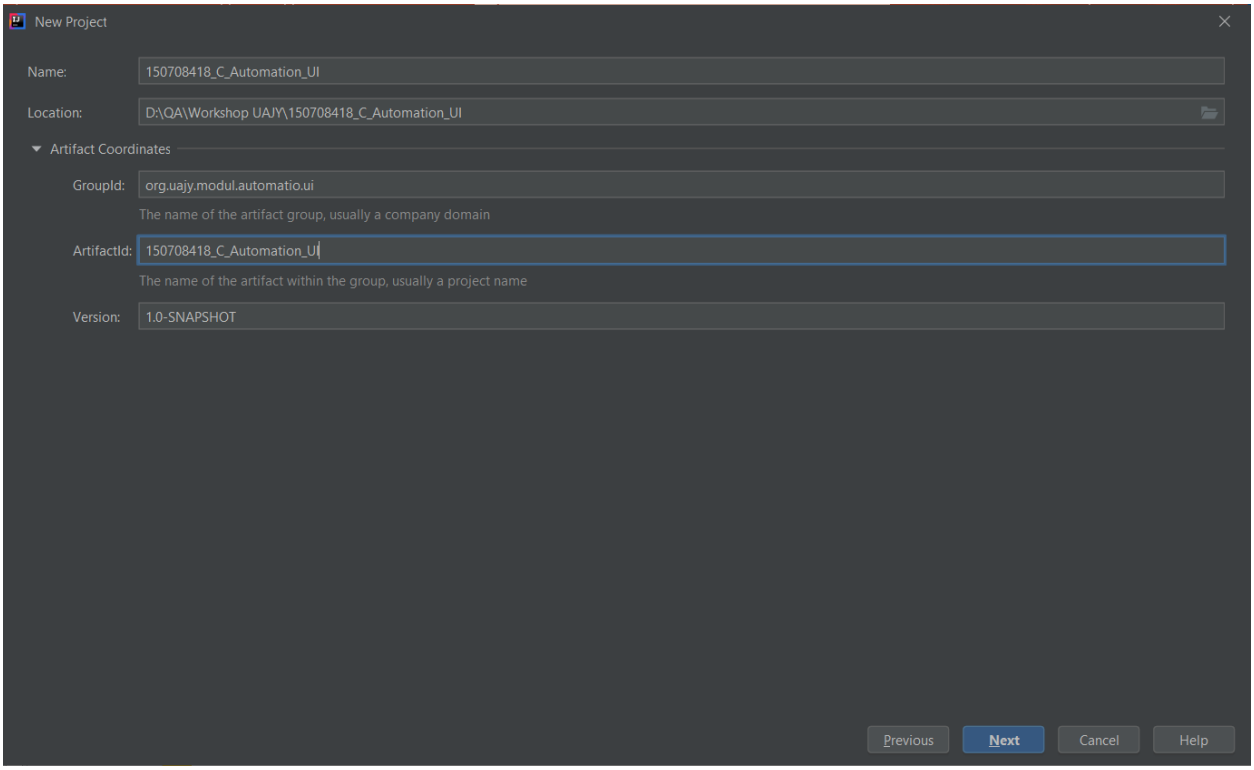
Berikut link untuk mendapatkan *archetype* ➔ https://mvnrepository.com/artifact/net.serenity-bdd/serenity-cucumber-archetype/2.0.81

```
<dependency>
    <groupId>net.serenity-bdd</groupId>
    <artifactId>serenity-cucumber-archetype</artifactId>
    <version>2.0.81</version>
</dependency>
```

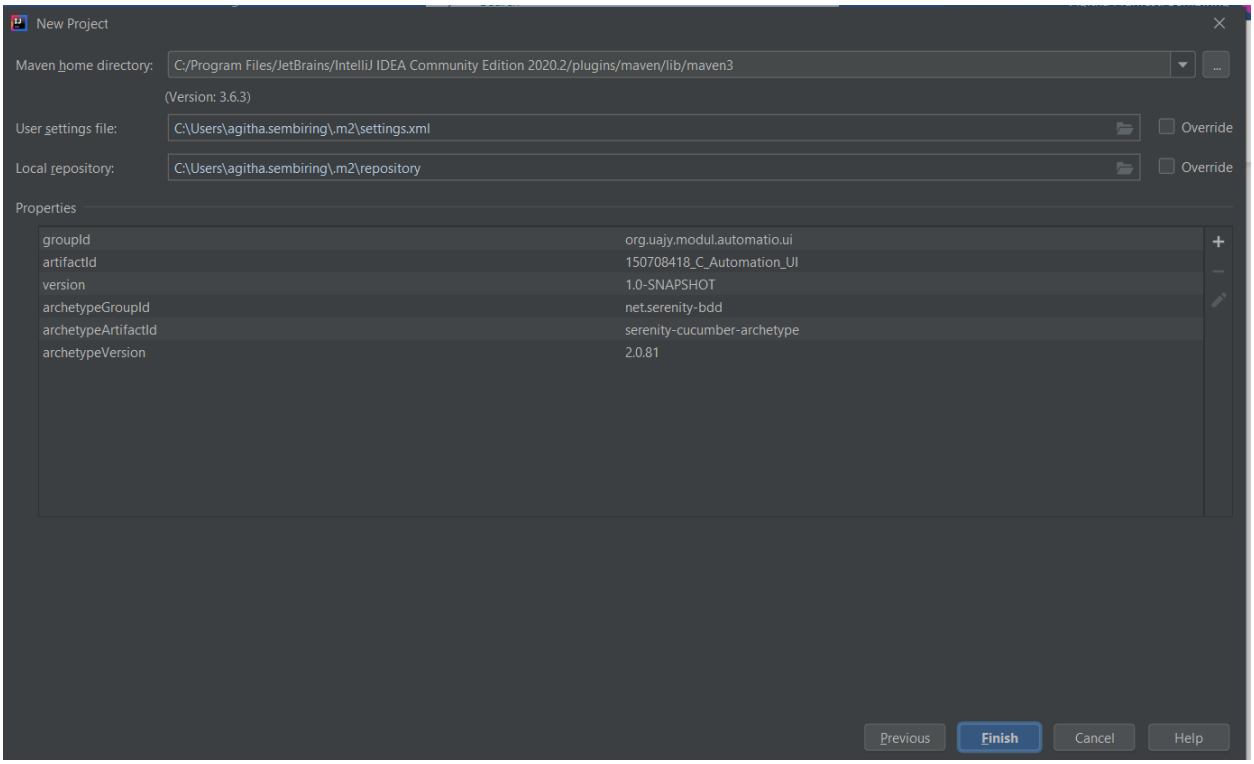
Masukkan *archetype* dan klik tombol '**OK**' dan klik tombol '**Next**':



4. Lalu pastikan lokasi folder sesuai dengan yang telah dibuat di **Langkah pertama.** Untuk name akan sama dengan artifactId menggunakan format 'NPM_Kelas_Automation_UI' dan groupId dapat menggunakan 'org.uajy.modul.automation.ui' lalu klik tombol '**Next**'

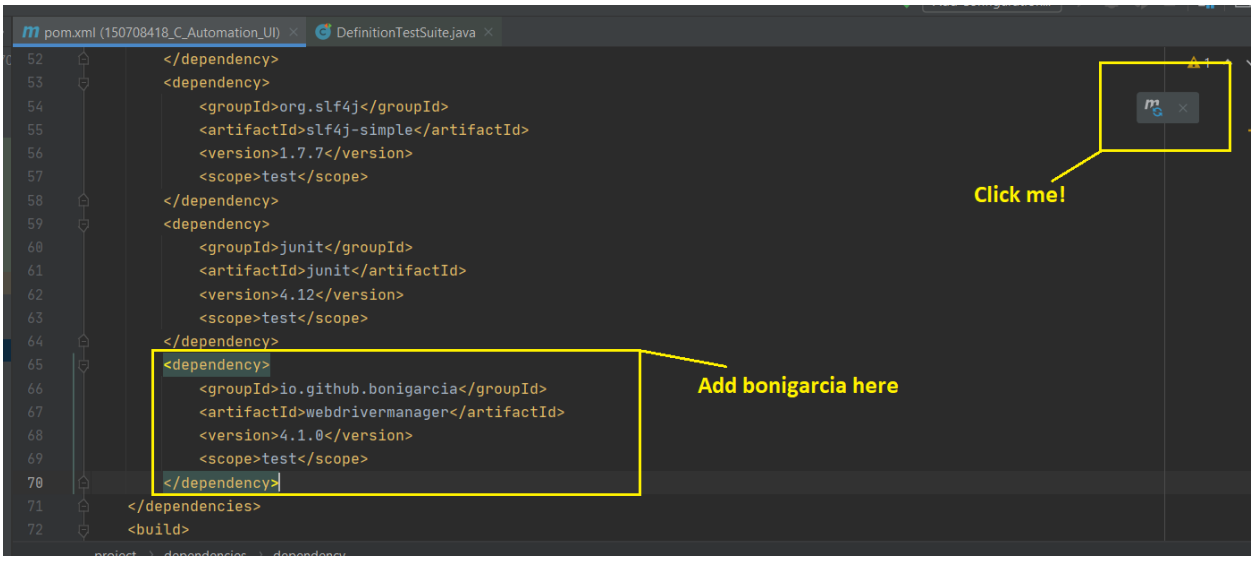


5. Lalu klik tombol '**Finish**'

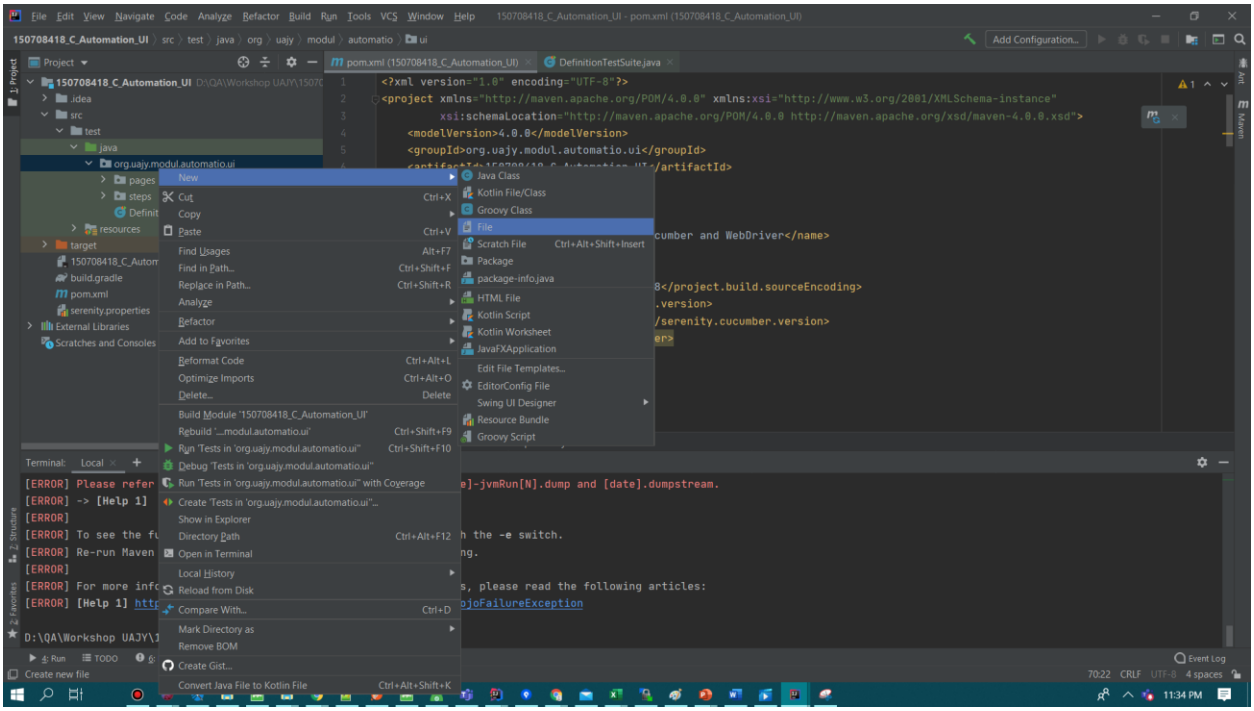


6. Karna kita akan mengimplementasikan Custom WebDriver, maka tambahkan *dependency* berikut di pom dan klik ikon kanan berwarna biru untuk men*download dependency* yang telah dimasukan ke pom:

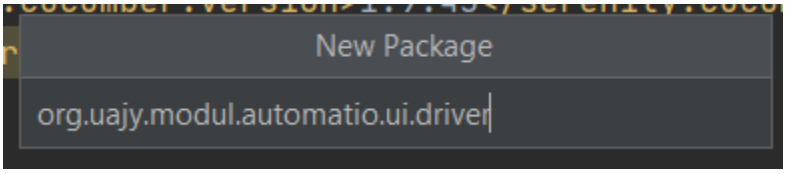
```
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>4.1.0</version>
    <scope>test</scope>
</dependency>
```



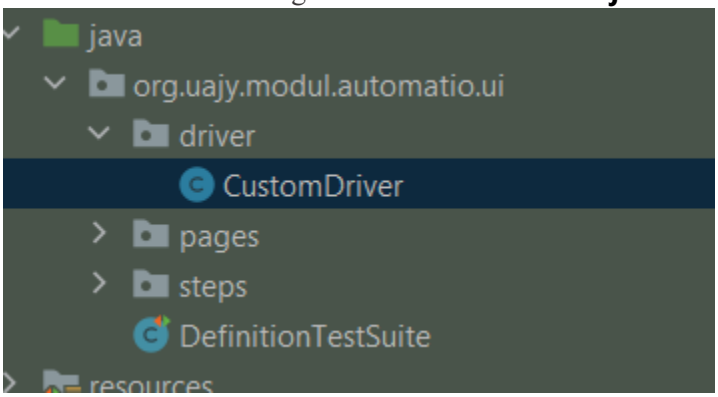
7. Lalu buat package dan kelas untuk Driver seperti berikut:



Beri nama *package*nya sebagai ‘driver’:



Lalu buat kelas Java dengan nama **CustomDriver.java** didalam package driver yang telah kita buat :



8. Lalu buka kelas CustomDriver.java dan masukkan settingan driver yang akan digunakan (disini kita menggunakan chrome sebagai browser yang akan menjalankan automation UI) :

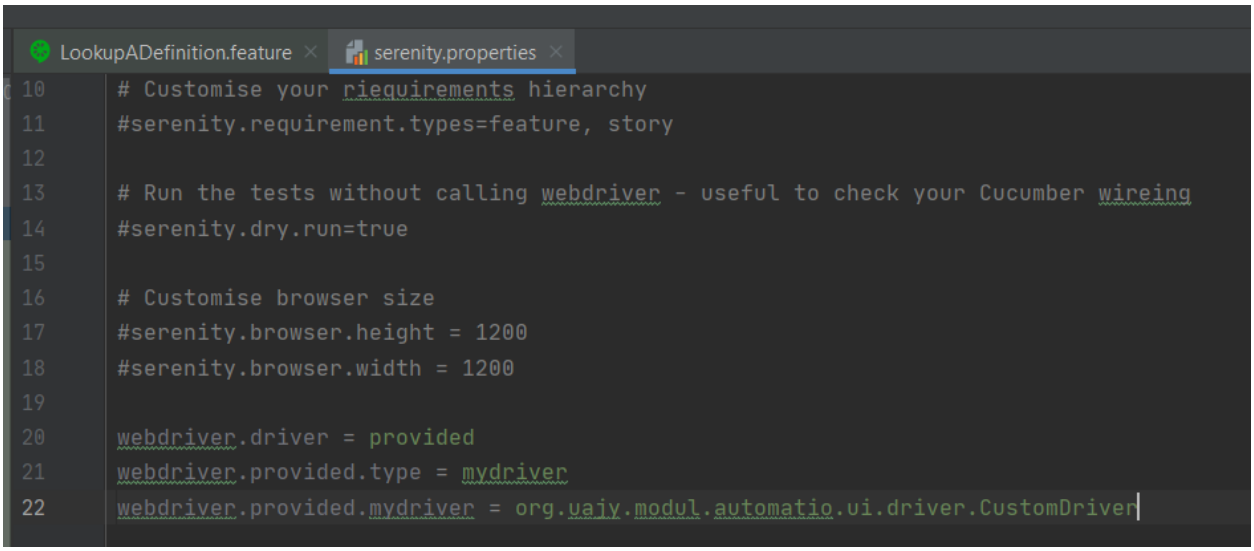
```
package org.uajy.modul.automatio.ui.driver;

import io.github.bonigarcia.wdm.WebDriverManager;
import net.thucydides.core.util.EnvironmentVariables;
import net.thucydides.core.util.SystemEnvironmentVariables;
import net.thucydides.core.webdriver.DriverSource;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class CustomDriver implements DriverSource {
    private EnvironmentVariables environmentVariables =
            SystemEnvironmentVariables.createEnvironmentVariables();


    @Override
    public WebDriver newDriver() {

        ChromeOptions options = new ChromeOptions();
        options.addArguments("start-maximized");
        WebDriverManager.chromedriver().setup();
        return new ChromeDriver(options);
    }

    @Override
    public boolean takesScreenshots() {
        return false;
    }
}
```

CustomDriver.java

```java
package org.uajy.modul.automatio.ui.driver;

import io.github.bonigarcia.wdm.WebDriverManager;
import net.thucydides.core.util.EnvironmentVariables;
import net.thucydides.core.util.SystemEnvironmentVariables;
import net.thucydides.core.webdriver.DriverSource;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;

public class CustomDriver implements DriverSource {
    private EnvironmentVariables environmentVariables =
            SystemEnvironmentVariables.createEnvironmentVariables();


    @Override
    public WebDriver newDriver() {

        ChromeOptions options = new ChromeOptions();
        options.addArguments("start-maximized");
        WebDriverManager.chromedriver().setup();
        return new ChromeDriver(options);
    }

    @Override
    public boolean takesScreenshots() {
        return false;
    }
}
```

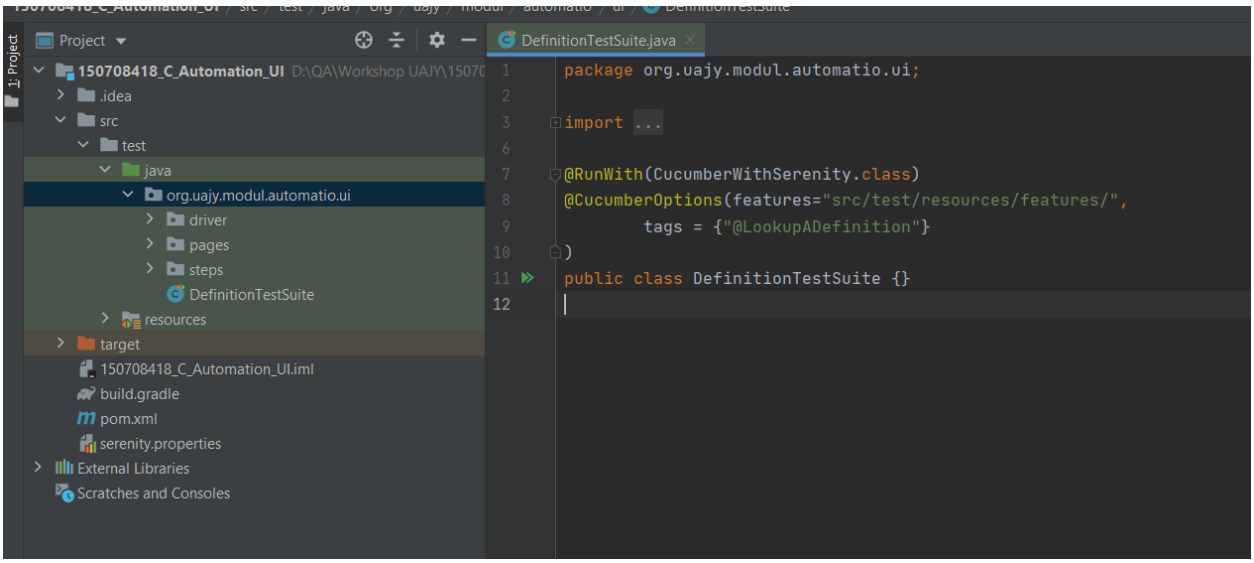
9. Selanjutnya tambahkan di serenity.properties untuk menyatakan bahwa kita akan menggunakan custom webdriver (alias bukan driver default):

```
webdriver.driver = provided
webdriver.provided.type = mydriver
webdriver.provided.mydriver = org.uajy.modul.automatio.ui.driver.CustomDriver
```

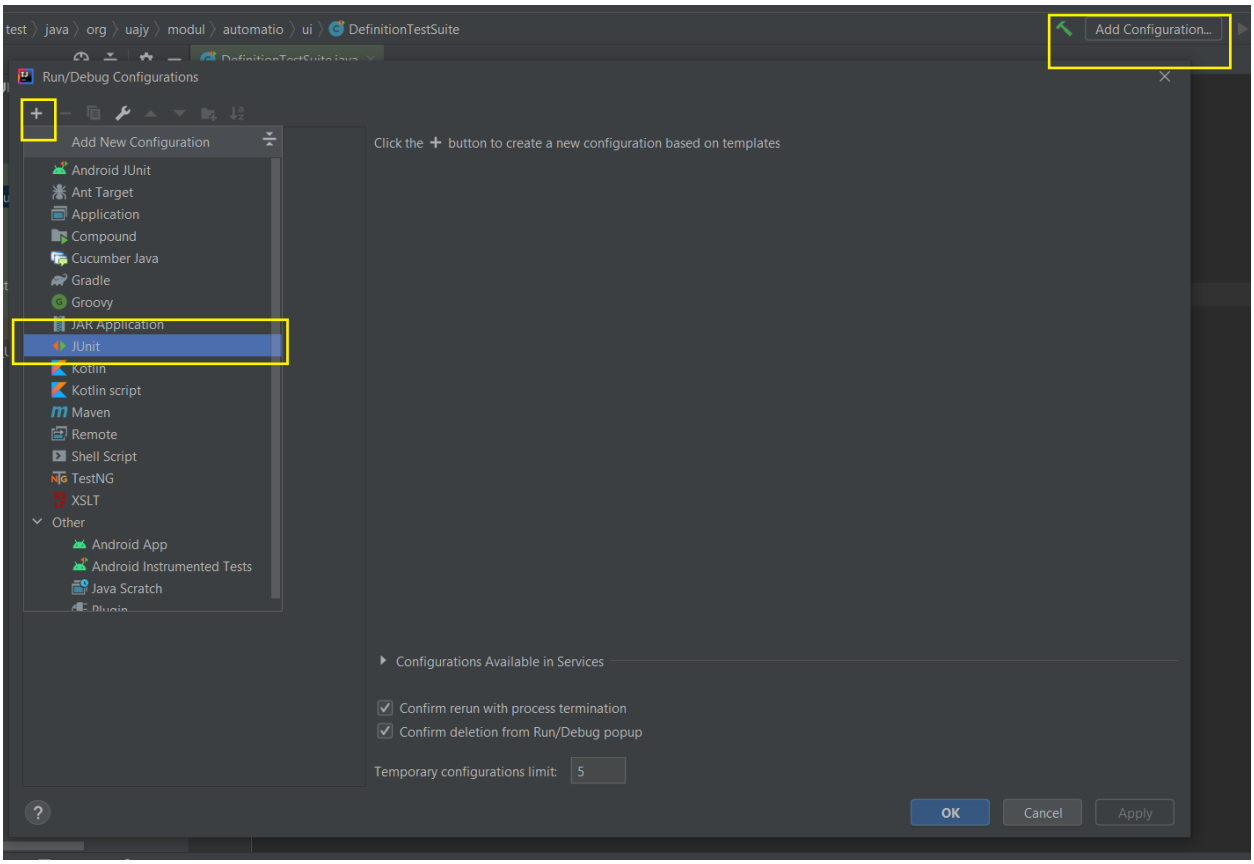


10. Lalu buka file 'DefinitionTestSuite' dimana ia merupakan cucumber runner dan lakukan perubahan seperti berikut:

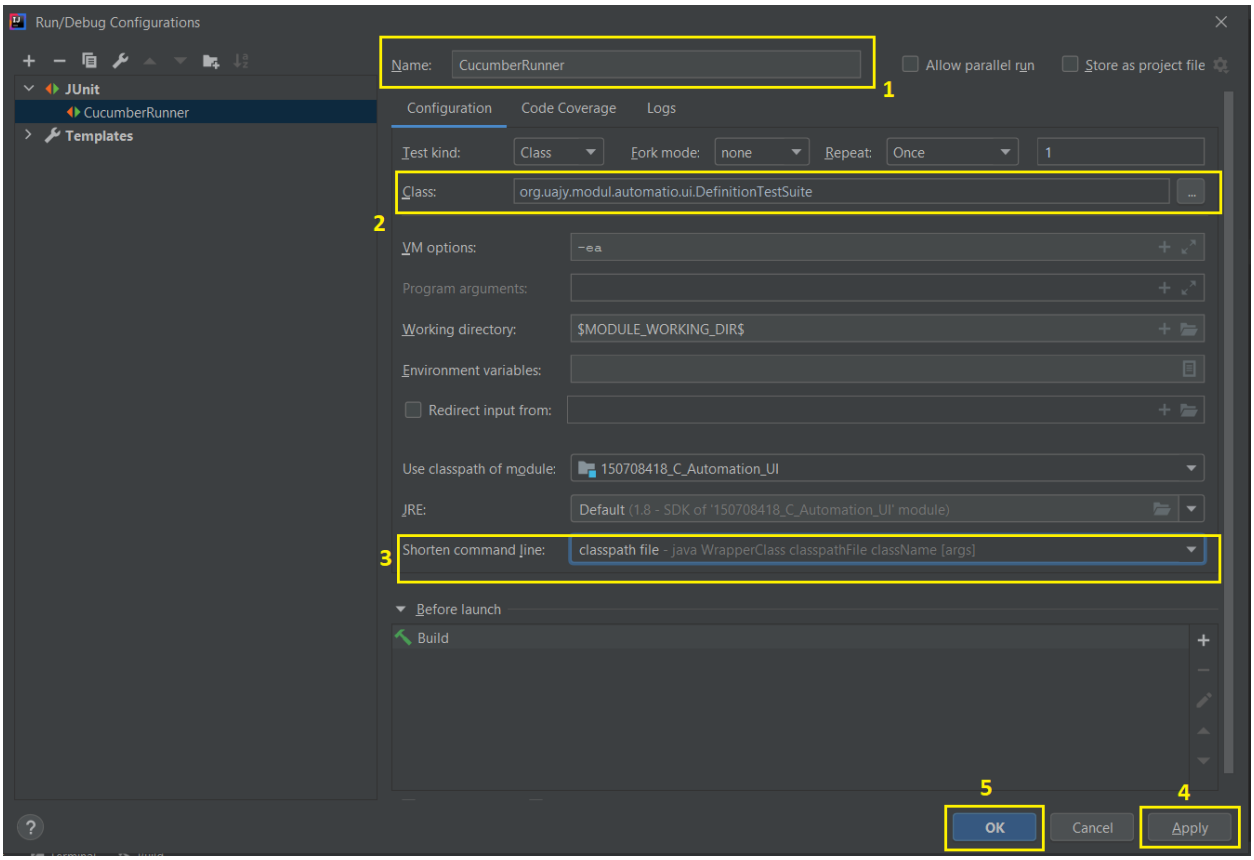
```
@RunWith(CucumberWithSerenity.class)
@CucumberOptions(features="src/test/resources/features/",
        tags = {"@LookupADefinition"}
)
public class DefinitionTestSuite {}
```



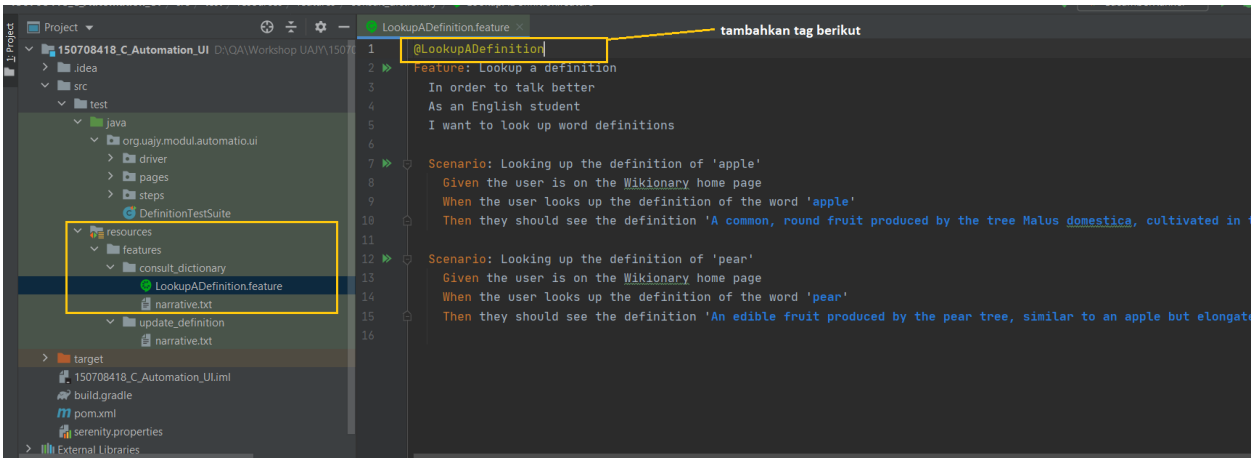
11. Lalu klik ‘Add Configuration’ ➔ ‘+’ ➔ ‘Junit’ :



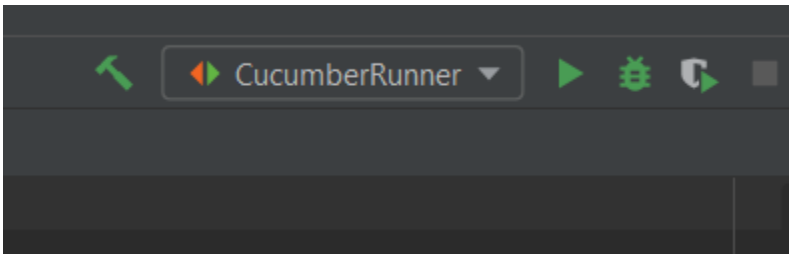
12. Isikan seperti berikut:



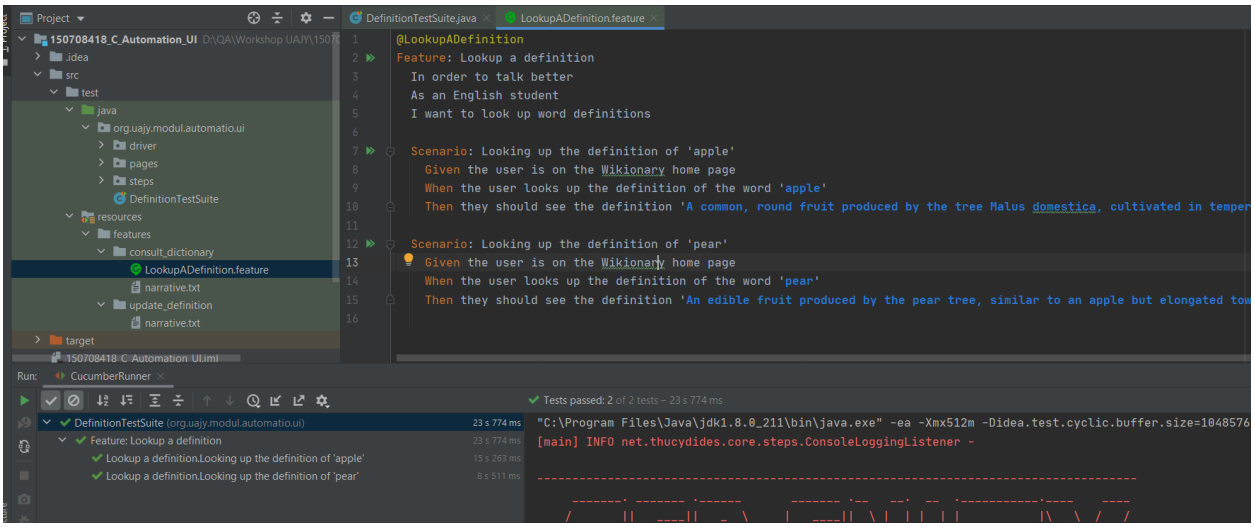
13. Tambahkan tag 'LookupADefinition' di file feature :



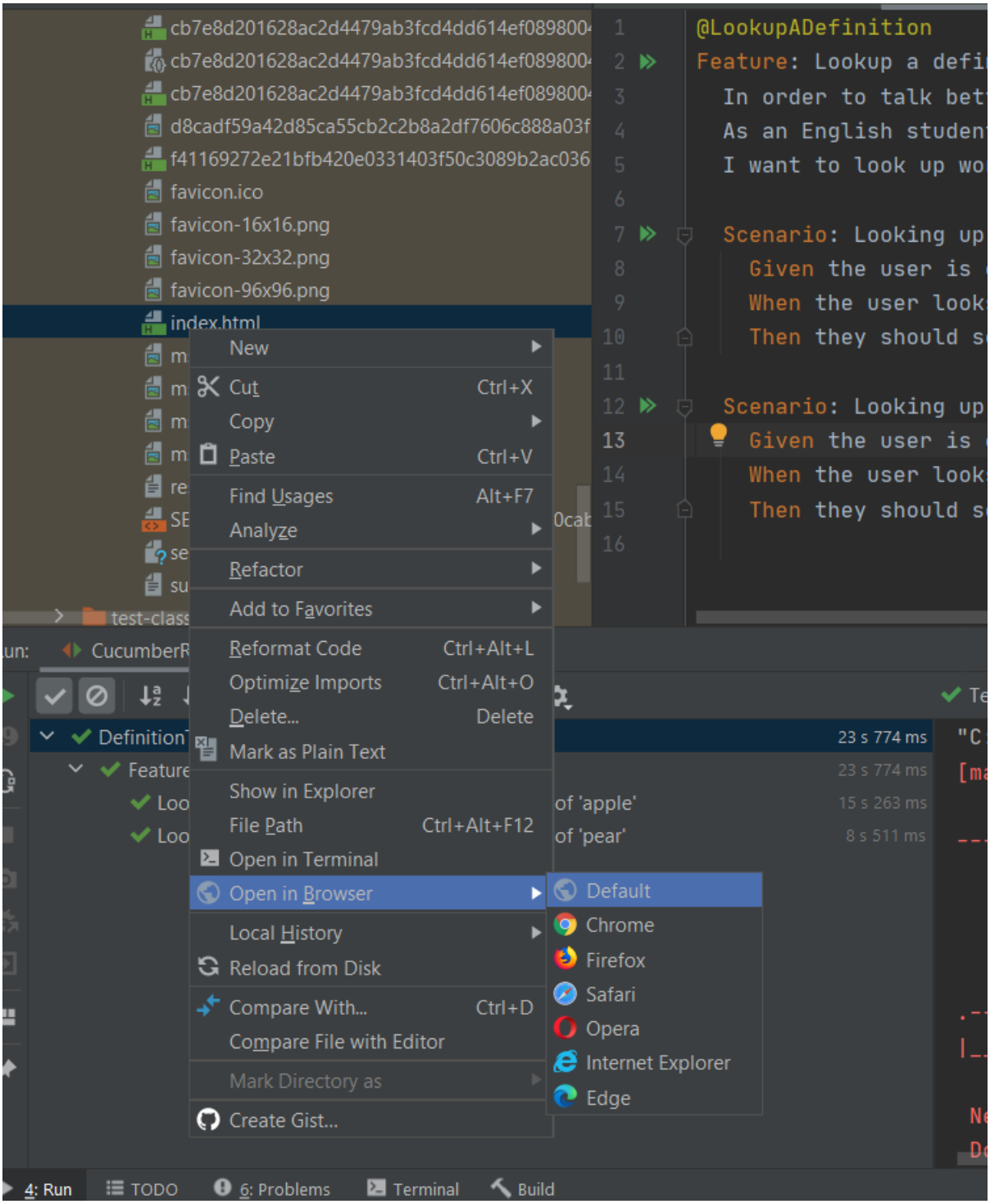
14. Lalu run dengan panah hijau berikut:

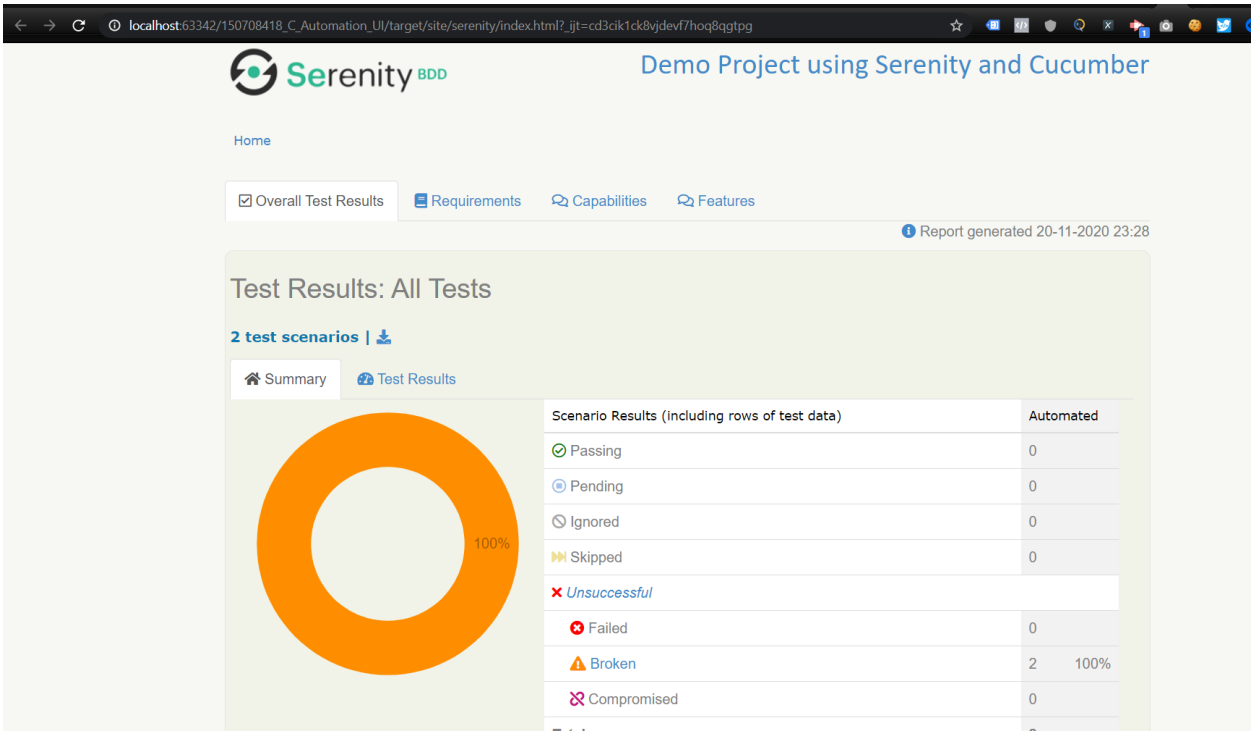


15. Result ➔ Success



16. Report : open target → open site → find index.html → klik kanan dan pilih browser untuk menampilkan report :





Task:

Let's create scenario for: http://gosoft.web.id/wonderfulQuote/ 😊