**Create and Install Cucumber Java:**

* Create Maven Project.
* Update the below jar for cucumber.

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-junit</artifactId>

<version>1.2.5</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-java</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-core</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>gherkin</artifactId>

<version>2.12.2</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>info.cukes</groupId>

<artifactId>cucumber-testng</artifactId>

<version>1.2.5</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.10.0</version>

</dependency>

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>6.11</version>

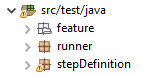
<scope>test</scope>

</dependency>

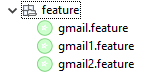
</dependencies>

* After updating maven dependencies, required 3 packages to implement BDD.

Feature, runner and stepDefinition.



* In feature package, create a file named as ‘test.feature’.



* In feature file, create BDD structure like

**Feature:** validate Gmail functionality

**Scenario:** verify Gmail functionality

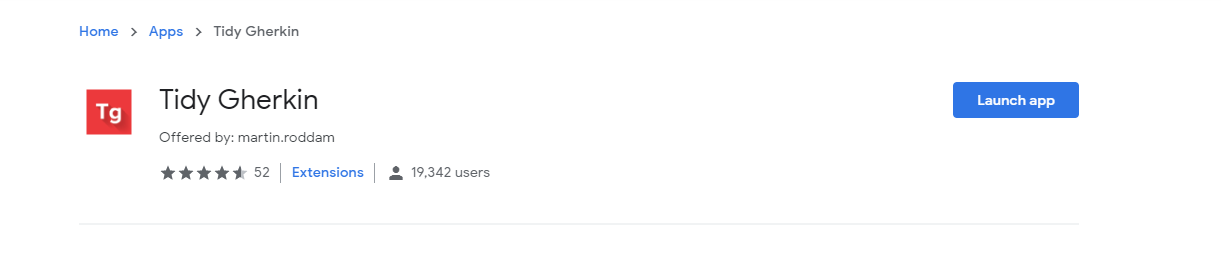
*Given* enter valid url for Gmail

*And* enter username and password

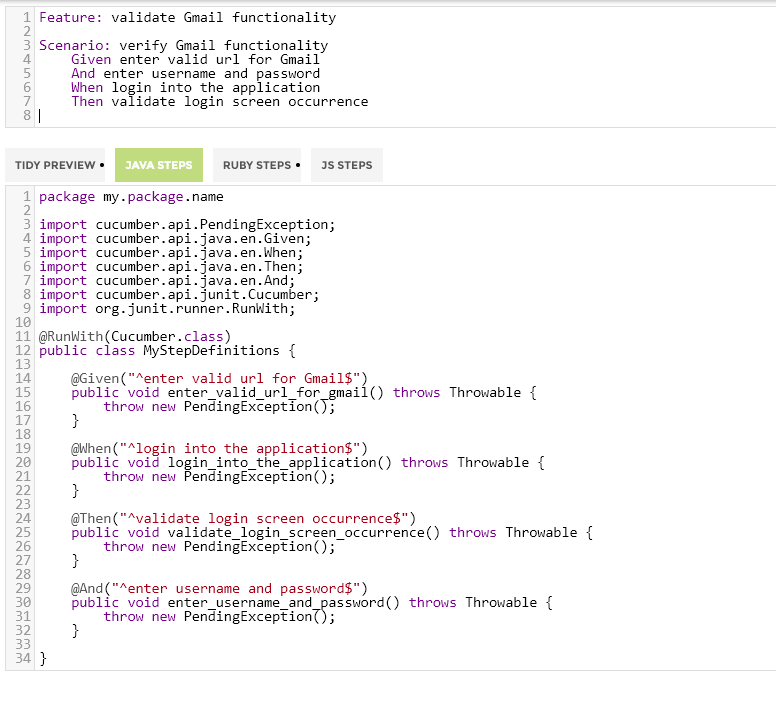
*When* login into the application

*Then* validate login screen occurrence

* Search for ‘Tidy Gherkins’ add-on in Google.



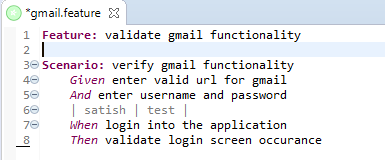
* Click on launch App
* Copy feature file then it will generate stepDefinition Code.

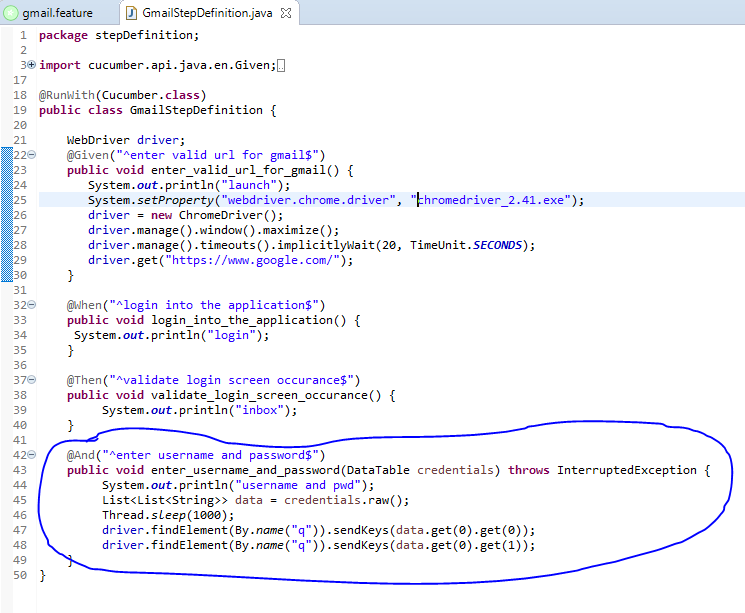


* Copy generated stepDefinition Code and paste it in stepdefinition class and do required modifications.

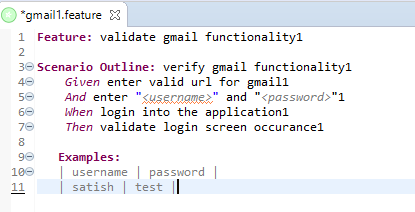
**Parameterized BDD, Related Code in Step Definition:**

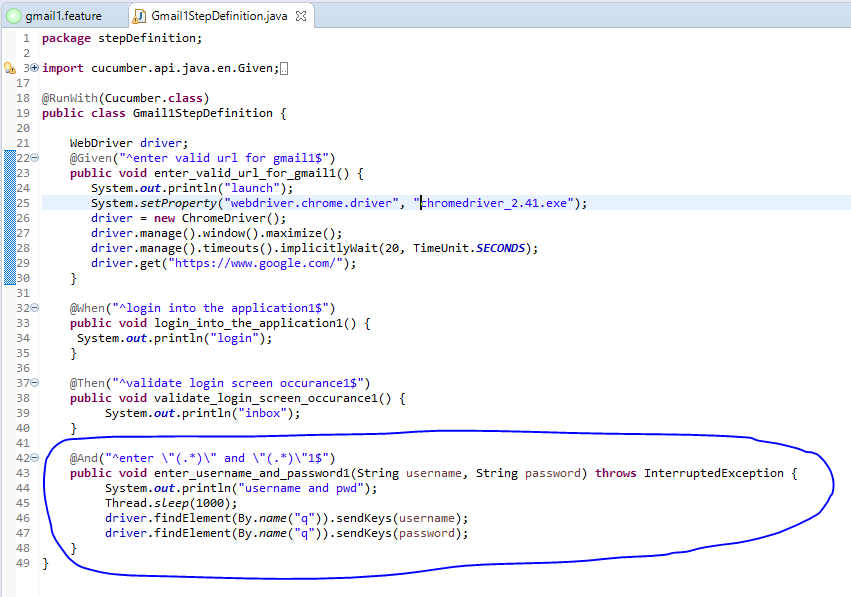
**1.**



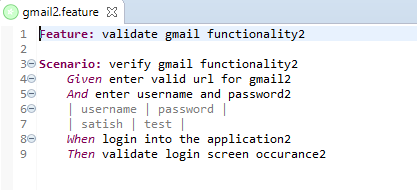


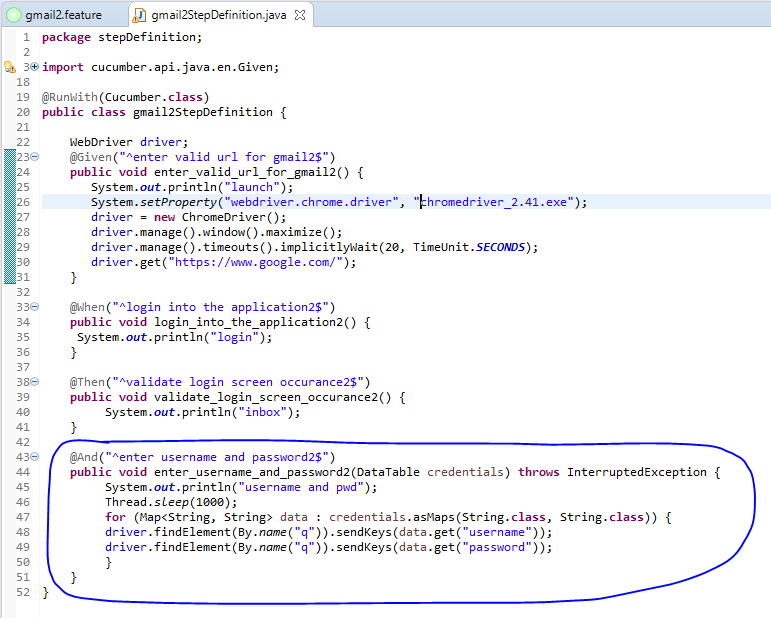
2.



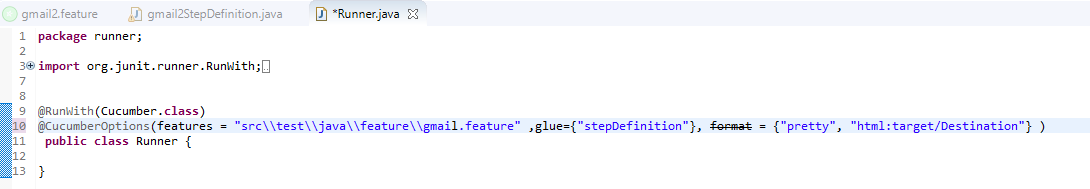


3.



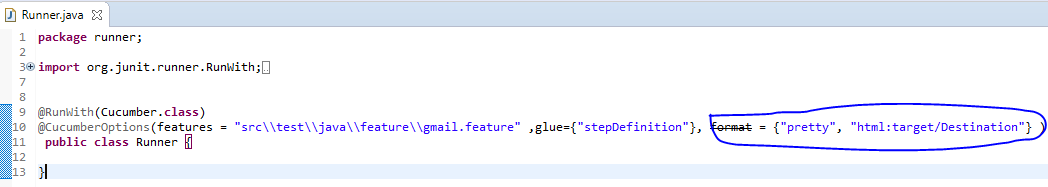


Runner File:



For generating HTML Reports, append format tag in cucumber options.

~~format~~ = {"pretty", "html:target/Destination"}

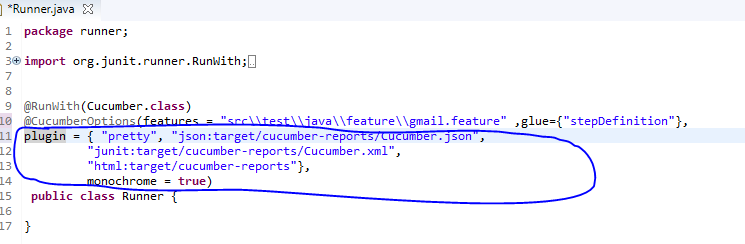


For generating All Type Reports (HTML, JSON, XML), append plugin tag in cucumber options.

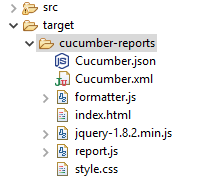
plugin = { "pretty", "json:target/cucumber-reports/Cucumber.json",

"junit:target/cucumber-reports/Cucumber.xml",

"html:target/cucumber-reports"},



Output Folder:



Cucumber.json 🡪 for json report.

Cucumber.xml 🡪 for xml report.

Index.html 🡪 for HTML report.

Monochrome = true 🡪 for generating report at console in readable format.

For generating extent reports in Cucumber refer <https://www.toolsqa.com/selenium-cucumber-framework/cucumber-extent-report/>.

**Cucumber Tags:**

**Feature:** validate gmail functionality

*@SmokeTest* *@RegressionTest*

**Scenario:** verify gmail functionality

*Given* enter valid url for gmail

*And* enter username and password

| satish | test |

*When* login into the application

*Then* validate login screen occurance

*@SmokeTest*

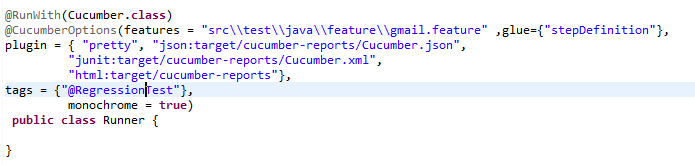
**Scenario:** verify login functionality

*Given* enter valid url for login

*@SmokeTest*

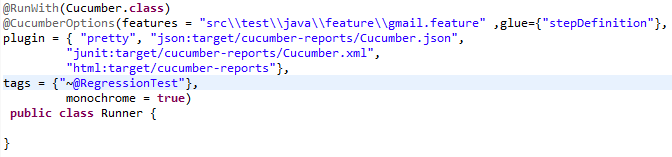
**Scenario:** verify login functionality

*Given* enter valid url for login



If we mention @RegressionTest in Runner class, it executes only Regression related scenarios.

If we mention @SmokeTest, Executes only Smoke related scenarios.



If we mention ~@RegressionTest in Runner class, it skips Regression related scenarios and execute remaining scenarios.

**Extent Report in Cucumber:**

* Add two dependencies for extent report generation in pom.xml file.

<dependency>

<groupId>com.aventstack</groupId>

<artifactId>extentreports</artifactId>

<version>3.0.6</version>

</dependency>

<dependency>

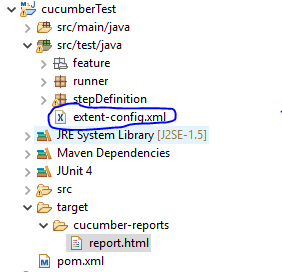
<groupId>com.vimalselvam</groupId>

<artifactId>cucumber-extentsreport</artifactId>

<version>3.0.2</version>

</dependency>

* Create one file named as ‘extent-config.xml’ under src/test/java package.



* Copy the below code in extent-config.xml file.

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

<extentreports>

<configuration>

<encoding>UTF-8</encoding>

<documentTitle>Cucumber Extent Reports - v1.0.0</documentTitle>

<reportName>Cucumber Extent Reports</reportName>

</configuration>

</extentreports>

* In runner class, update plugin as

plugin = {"com.cucumber.listener.ExtentCucumberFormatter:target/cucumber-reports/report.html"},

* Update the code under runner class

**public** **class** Runner {

@AfterClass

**public** **static** **void** setup()

{

Reporter.*loadXMLConfig*(**new** File("src\\test\\java\\extent-config.xml"));

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Optional \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

Reporter.*setSystemInfo*("User Name", "AJ");

Reporter.*setSystemInfo*("Application Name", "Test App ");

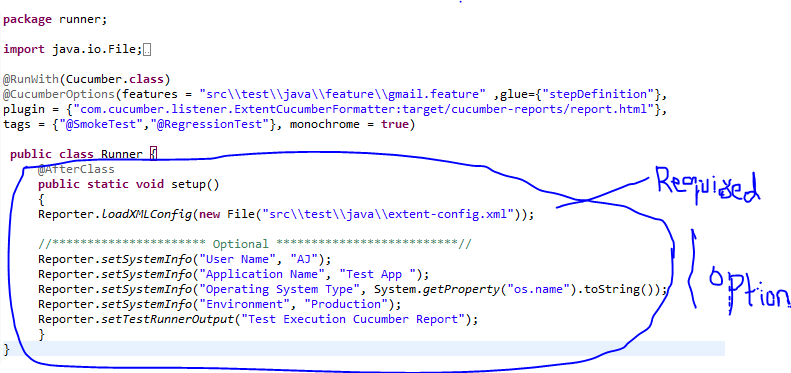
Reporter.*setSystemInfo*("Operating System Type", System.*getProperty*("os.name").toString());

Reporter.*setSystemInfo*("Environment", "Production");

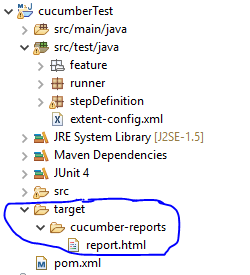
Reporter.*setTestRunnerOutput*("Test Execution Cucumber Report");

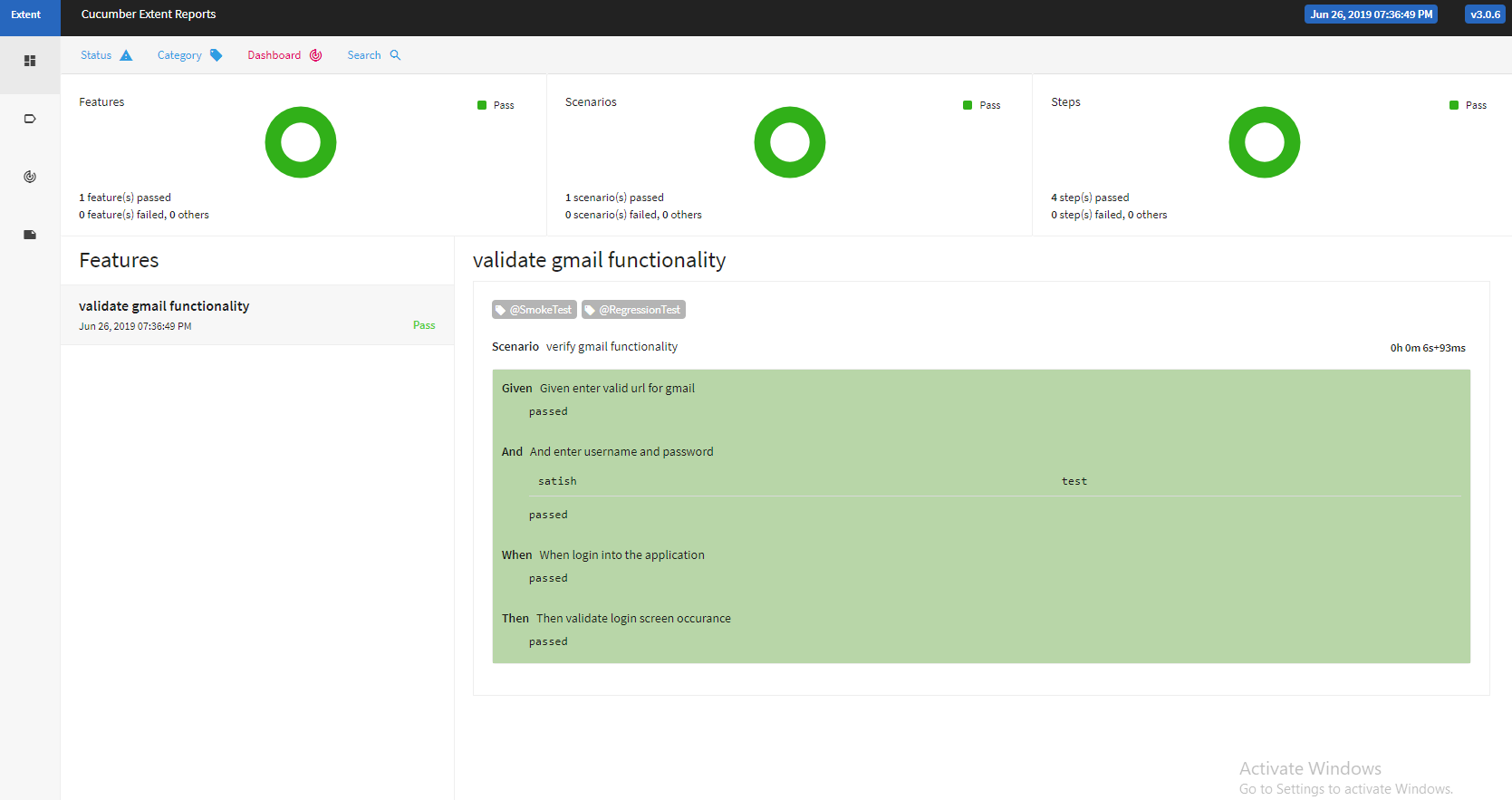
}

}



Note: All Annotations should be junit not TestNG.

* Run as junit Run.
* Finally generate extent report under ‘target🡪cucumber-reports🡪reports.html’
* 



For optional parameters in @AfterClass

