### **Cucumber Terminologies -**

1. Feature file
2. Step Definition layer
3. Runner file

### **Cucumber**

* Cucumber is a testing framework which supports Behavior Driven Development (BDD).
* It defines application behavior in plain meaningful English text using a simple grammar defined by a language called Gherkin.
* *Cucumber is written in Ruby, but it can be used to write “test” code in Ruby or other languages like java, C#, Python.*

***What is Gherkin?***

* The language used to create feature files is known as **Gherkin**.
* It is ***Business readable domain specific language***(*BRDSL*) and a language **defines application behavior in plain meaningful English text.**

Below are keywords defined by ***Gherkin***.

1. ***Feature***
2. ***Scenario***
3. ***Given***
4. ***When***
5. ***Then***
6. ***And***
7. ***But***

***Other Keywords***

1. ***Scenario Outline***
2. ***Rule***
3. ***Background***
4. ***' \* '***
5. ***Feature:*** 
   1. Feature defines the logical test functionality that will test in feature file. summary of what you will be testing.
   2. Everything after *Feature*: till the next Keyword is considered as feature description.
   3. *Description is not a keyword of Gherkin*.
6. ***Scenario:***
   1. Each Feature will contain a number of tests . Each test is called a ***Scenario***
   2. Each scenario/test can be basically broken down into three parts:
      1. ***Precondition*** *to the test, which represent with (Given) keyword*
      2. ***Test step*** *execution, which represent with (When) keyword*
      3. ***Verification*** *of the output with expected result, which represent with (Then)*
7. ***Given:***
   1. ***Given*** defines a precondition to the test.
8. ***When:*** 
   1. ***When*** keyword defines the test action that will be executed.
9. ***Then:***
   1. ***Then*** keyword defines the Outcome of previous steps, in which ***we match*** *actual with expected result.*
10. ***And:***
    1. ***And*** keyword is used to add conditions to your steps (If extra steps to test we can add it to Given, When and Then)
11. ***But:***
    1. ***But*** keyword is used to add negative type of conditions.

| ***Feature****: LogIn Test*  *Description: This feature will test a LogIn and LogOut functionality* |
| --- |
| ***Scenario****: Successful Login with Valid Credentials*  ***Given*** *User is on Home Page*  ***When*** *User Navigate to LogIn Page*  ***And*** *User enters UserName and Password*  ***Then*** *Message displayed with successful login* |
| ***Scenario****: Unsuccessful Login with InValid Credentials*  ***Given*** *User is on Home Page*  ***When*** *User Navigate to LogIn Page*  ***And*** *User enters UserName and Password*  ***But*** *The user credentials are wrong*  ***Then*** *Message displayed Wrong UserName & Password* |

***Rule*** is meant to cover one or more scenarios that explain the particular rule.

## 

## **Background: Keyword**

***Background*** keyword is used to define steps that are common to all the tests in the feature file. For example, to purchase a product, you need to do the following steps:

* *Navigate to Home Page*
* *Click on the LogIn link*
* *Enter UserName and Password*
* *Click on Submit button*

## **It works in three layer**

1. Feature Layer
2. Step Defi*nitions*
3. Cucumber Options

## **What is Cucumber Feature File?**

* A feature is a functionality or standalone unit of a software application.
* the feature is a parameter which is used to test the requirements of the customer.
* Eg.

| * Create and delete the user from the social networking site. * User login functionality to access the social networking site. * Sharing videos or photos on the social networking site. * Sending a friend request. * Logout or sign out. |
| --- |

* It is a best practice later when we start testing, before deriving the test scripts, we should write the features to be tested.
* A file in which we store features, description about the features and scenarios to be tested is known as **Feature File**.
  + A feature file contains a list of scenarios to be tested for that feature.

A simple feature file consists of the following keywords/parts −

* **Feature** − Name of the feature under test.
* **Description** (optional) − Describe about feature under test.
* **Scenario** − What is the test scenario.
* **Given** − Prerequisite before the test steps get executed.
* **When** − test action that will be executed.
* **Then** −Outcome / result

## Steps Definitions

* A Step Definition is a java class which will have implementation of all steps from feature file for each scenario.
* Each step is implemented with annotation of Gherkin keywords.
* Each *Step Definition* linked to all the matching *Steps*.
* So to execute feature we have to write the code in step definition file and run it from Cucumber Options.

## Cucumber **Options**

1. ***@CucumberOptions*** is a property file or setting for run tests.
2. It is created with test runner class.
3. The test runner class acts as interlink between feature files and step definition classes.
4. In test runner class,we can provide the path for both feature file and step defs class with a test runner class, we have the option to run either a single feature file, or multiple feature files as well.

## 

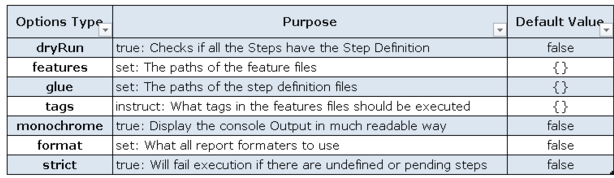
## Understanding the annotations in the test runner class

## These are two annotations as **@RunWith** and **@CucumberOptions**

**@RunWith annotation:** This is JUnit annotation that specifies which runner it has to use to execute this class. You can see that we have to provide **Cucumber.class** as a parameter with this.

**@CucumberOptions annotation:** This annotation provides some important information which will be used to run your cucumber feature file

1. **Features**
   1. **Features Options** helps to locate the Feature file in the project folder structure.
   2. All we need to do is to specify the folder path and Cucumber will automatically find all the '.features' extension files in the folder
      1. **features = "src/test/java/features"**
2. **Glue**
   1. Glue Options helps to locate the **Step Definition file** in the project folder structure.
   2. All we need to do is to specify the folder path and Cucumber will automatically find all class from package
      1. **glue = "src/test/java/stepDefinition"**
3. **Monochrome**
   1. This option can either set as **true** or **false**.
   2. If it is set as true,then the console outputs are more readable
   3. if it is set as false, then the console output is not as readable
      1. **monochrome=true/false**
4. **Dry Run**
   1. **This** option can either set as **true** or **false by default is false**.
   2. If it is set as true, it means Cucumber will only check that every Step in the Feature File has corresponding code implementation in Step Definition file or not.
      1. druRun=true/false
5. ***Fo*rmat**
   1. **Format Option** is used to specify different formatting options for the output reports.
   2. **Pretty**: Prints the Gherkin source with additional colors and stack traces for errors.
      1. **plugin= {"pretty"}**
   3. **HTML:** This will generate a HTML report at the location mentioned in the for-matter itself.
      1. **plugin= {"html:Folder\_Name/cucumber.html"}**
   4. **JSON**: This report contains all the information from the gherkin source in JSON Format.
      1. **plugin= {"json:Folder\_Name/cucumber.json"}**
   5. **JUnit**: This report generates XML files
      1. **format = { "junit:Folder\_Name/cucumber.xml"}**

****