

## Cucumber IQ:

### 1. What is Cucumber? BDD vs TDD? Why did you decide to start using Cucumber ? What is the difference between Cucumber and TestNG?

**Cucumber** is a tool used to run automated acceptance tests created in a BDD format. One of its most outstanding features of the tool is the ability to carry out plain-text functional descriptions (written in the language called Gherkin) as automated tests.

**BDD** exactly means Explaining the customer requirement in plain text. In BDD, users (business analysts, product owners) first write scenarios or acceptance tests that describes the behavior of the system from the customer's perspective, for review and sign-off by the product owners before developers write their codes.

#### **BDD vs TDD**

**BDD** is in a more readable format by every stake holder since it is in English, unlike **TDD** test cases written in programming languages such as Ruby, Java etc.

**BDD** explains the behavior of an application for the end user while **TDD** focuses on how functionality is implemented. Changes on functionality can be accommodated with less impact in **BDD** as opposed to **TDD**.

**BDD** enables all the stakeholders to be on the same page with requirements which makes acceptance easy, as opposed to **TDD**.

**The best reason for choosing the cucumber is that we can explain requirement in simple english.** Cucumber software tool allows the stakeholders of business to be involved in it even if they don't know how to code. The experience of the users is prioritized by using the Cucumber tool. Cucumber tool allows the codes to be reused.

#### **Difference between TestNG and Cucumber**

**TestNG** is a testing framework where you specify test classes that needs to be executed.

**Cucumber** is to run acceptance tests written in Behaviour driven development style. Gherkin is the language that Cucumber uses to define test cases.

## 2. How did you use cucumber in your framework?

Main components of cucumber are **step definitions** and **feature files**. Feature files will have test cases which are written in **Gherkin** Language using **Given When Then** and Step definitions we are writing actual automation code using **Selenium and Java**. All the Step Definitions are stored in a separate folder with naming convention as **src/test/java** and all the feature files are stored in a separate folder with naming convention as **src/test/resources**. And then we will have a **TestRunner** class that uses **JUnit** to run our tests. TestRunner is starting point of test execution which gathers all tests to be executed.

## 3. Give an example of behavior driven test in plain text?

**Feature:** Defines what feature you will be testing in the tests below

**Scenario:** we are describing the test case here

**Given:** Tells the precondition of the test

**When** Defines actions to be taken

**And:** Defines additional/ continues conditions of the test

**Then:** States the post condition. You can say that it is expected result of the test.

### Example

#Author: syntax team

**Feature:** Login

#### **Background:**

**Given** I see OrangeHrm logo is displayed

#### **@smoke**

**Scenario:** Login with valid credentials

**When** I enter a valid username and password

**And** I click on the login button

**Then** I successfully logged in

## 4. What are the two files required to run a cucumber test?

- A feature file.
- A step definition file.

**5. What is a feature file? What is the extension? What is there is feature file? What are the keywords used in feature file?**

A **Feature File** is an entry point to the Cucumber tests. This is a file where we describe our 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

**.feature** is the extension of feature file. Example, “Test.feature” is a feature file name.

In the feature file we will define below keywords:

**Feature, Scenario, Background, Scenario Outline, Given, When, And ,But and Then.** These are keywords defined by Gherkin

**6. What is the language used to write scenarios in feature file? What is the main purpose?**

Gherkin is a plain-text language used to write cucumber scenario in a feature file. Gherkin is a business readable, domain-specific language used to describe the behavior of software applications from user’s perspective so that it's easy for non-programmers to read and collaborate.

Gherkin is not necessarily used to write automated tests. Gherkin is primarily used to write structured tests which can later be used as project documentation. The property of being structured gives us the ability to automate them. This automation is done by Cucumber

**7. Is it mandatory to use Given, When, Then keywords while writing scenario? What is the difference?**

No it is **not mandatory to use Gherkin keywords**. Gherkin keywords makes our scenario be in a more readable format. \* can also be used to write steps in the feature file.

**Feature:** Account Balance

**Scenario:** Verify Positive Balance

\* I have \$100 in my account

- \* I withdraw \$50
- \* I should have \$50 balance

**8. Name any 2 testing framework that can be integrated with Cucumber?**

- TestNG
- Junit

**9. Name any two build management tools that can be integrated with Cucumber?**

- Gradle
- Maven

**10. Explain Cucumber Tags? How to run only specific scenarios? How to run multiple scenarios with tags? How to exclude tagged scenarios?**

**Cucumber tags** are used to organize scenarios in feature file. We can have as many tags as we like before a scenario or feature. @ is used to represent tags. Example: @regression, @sprint5, @EndtoEnd

Tags are used to:

- Group scenarios
- Ignore scenarios from execution
- Logically group (OR & AND)

For example we are having feature file with three test cases here i want to execute only smoke test case:

**Feature:** ECommerce Application

**@SmokeTest**

**Scenario:** Successful Login

**Given** This is a blank test

**@RegressionTest**

**Scenario:** UnSuccessful Login

**Given** This is a blank test

**@SmokeTest**

**Scenario:** Add a product to bag

**Given** This is a blank test

To execute **specific tags** we have to mention them in CucumberOption in our runner class.

**@Cucumber.Options(tags={"@SmokeTest"})**

To execute **multiple tags**:

Execute all tests tagged as @SmokeTest **OR** @RegressionTest  
Tags which are comma separated are ORed.

**@CucumberOptions(tags={"@SmokeTest,@RegressionTest"})**

Execute all tests tagged as @SmokeTest **AND** @RegressionTest  
Tags which are passed in separate quotes are ANDed

**@CucumberOptions(tags={"@SmokeTest","@RegressionTest"})**

**to skip specific tags**

Special Character ~ is used to skip the tags.

Execute all tests of the feature tagged as @RegressionTest but skip scenarios tagged as @SmokeTest

**@CucumberOptions(tags={"~@SmokeTest","@RegressionTest"})**

This is AND condition, which means all the scenario tagged as @FunctionalTest but not @SmokeTest

## **11. What is the use of keyword "background" in feature file?**

Background in Cucumber is used to define a step or series of steps which are common to all the tests in the feature file. It allows us to add some context to the scenarios for a feature where it is defined. A Background is much like a scenario containing a number of steps. But it **runs before each and every scenario where for a feature in which it is defined.**

**#Author:** SyntaxTeam

**@Employee**

**Feature:** Employee

**Background:**

**Given** I logged into OrangeHRM

**@Smoke**

**Scenario Outline:** Add Employee

**And** I navigated to the Add Employee Page

**When** I provide "<firstName>" and "<middleName>" and "<lastName>"

**Then** I successfully added an employee

**Examples:**

firstName	middleName	lastName	
Alex	S	Smith	
Jane	H	Han	
Michael	J	Jackson	

## 12. Explain Cucumber Hooks? What's Before/After hook?

**Cucumber Hooks** are blocks of code that can be used to run before and after the scenarios using `@before` and `@after` methods. It helps us eliminate the redundant code steps that we write for every scenario and also manages our code workflow.

```
public class Hooks {
```

**@Before**

```
public void start() {  
    BaseClass.setUp();  
}
```

**@After**

```
public void end(Scenario scenario) {  
    //if scenario fails  
    if(scenario.isFailed()) {  
        //take screenshot  
        TakesScreenshot pic=(TakesScreenshot)BaseClass.driver;
```

```
byte[] picture = pic.getScreenshotAs(OutputType.BYTES);
//attach to the report
scenario.embed(picture, "image/png");
}
BaseClass.tearDown();
}
}
```

**Note:** @After will be executed even though the scenario is failed. @Before will be executed before 'Background' steps in the feature file.

### 13. What is @CucumberOptions in test runner? List the properties of @CucumberOptions

@CucumberOptions are used to set specific properties for our cucumber test. Properties are:

- **Feature** – path to feature file
- **Glue** – path to step definition
- **dryRun** – boolean value – check for missing step definition
- **tags** – used to group cucumber scenarios in the feature file
- **plugin** – What all report formats to use
- **monochrome** – boolean value – display console output in a readable way

@RunWith(Cucumber.class)

```
@CucumberOptions(features= {"src/test/resources/features"}
, glue= {"com/syntax/stepDefinitions"}
, plugin= {
    "pretty",
    "html:target/cucumber-default-reports",
    "json:target/cucumber.json"
}
, tags= {"@Login"}
, monochrome =true
)
```

```
public class TestRunner {
```

}

#### 14. What is the difference between scenario and scenario outline?

##### Scenario

Represents a particular functionality which is under test. By seeing the scenario user should be able to understand the intent behind the scenario and what the test is all about.

##### Scenario Outline

Keyword in feature file is used to execute scenarios multiple times using a different set of test data. Multiple sets of test data are provided by using 'Examples' in a tabular structure separated by pipes (|). We can achieve data driven testing using scenario Outline

**Feature:** Add Employee

**Background:**

**Given** I logged in into OrangeHrm

**And** I click on PIM link

**And** I click on Add Employee link

@smoke @addEmployee

**Scenario Outline:** Add Employee

**When** I provide "<FirstName>", "<MiddleName>", "<LastName>" and "<Location>"

**And** I click on the save button

**Then** I see "<FirstName>", "<LastName>" is displayed

**Examples:**

FirstName	MiddleName	LastName	Location
Jane	J	Smith	Smart Office
James	S	West	Tech Office
Jackie	N	Eats	HQ

#### 15. How can we achieve data driven testing in Cucumber? What is DataTable in Cucumber? DataTable vs Scenario Outline?

To achieve data driver nesting in Cucumber we can use:

- Scenario Outline and Examples keyword



- DataTable

Cucumber DataTables let us store data in feature file.

There are 2 types of DataTable: with the header and without the header.

### Cucumber DataTable without header

#Author: syntax team

@addEmployee @sprint5

**Feature:** Add Employee

#### Background:

**Given** I logged in into OrangeHrm

**And** I click on PIM link

**And** I click on Add Employee link

#### @regression

**Scenario:** Add Employee Labels Verification

**Then** I see following labels

First Names	
Middle Name	
Last Name	
Employee Id	
Location	
Create Login Details	

### Implementation in Step Definitions

@Then("I see following labels")

public void i\_see\_following\_labels(DataTable addEmpLabels) {

System.out.println("----Expected labels from cucumber dataTable----");

List<String> expectedLabels = addEmpLabels.asList();

for (String label : expectedLabels) {

System.out.println(label);

}

}

### Cucumber DataTable with header

#author: john.smith@wellfargo.com

@sprint3 @login

**Feature:** Login

**Background:**

**Given** I see OrangeHrm logo

**@regression**

**Scenario:** Error message Validation

**When** I enter invalid username and password I see errorMessage

UserName	Password	ErrorMessage
Admin	Admin123	Invalid Credentials
Admin1	Syntax123!	Invalid Credentials
Admin2	Syntax123	Invalid Credentials

**Step Definition File:**

**@When("I enter invalid username and password I see errorMessage")**

**public void i\_enter\_invalid\_username\_and\_password\_I\_see\_errorMessage(DataTable wrongCredentials) {**

**List<Map<String, String>> maps = wrongCredentials.asMaps();**

**for(Map<String, String> map: maps) {**

**//passing username and password values from datatable/map**

**sendText(login.userName, map.get("UserName"));**

**sendText(login.password, map.get("Password"));**

**click(login.loginBtn);**

**//verifying text of error message**

**String actualError=login.message.getText().trim();**

**String expectedError=map.get("ErrorMessage");**

**Assert.assertEquals(expectedError, actualError);**

**}**

**}**

**NOTE:**Data from datatable can be retrieved as a single map (if we have only 1 line of data) or as a List of Maps:

## **DataTable vs Scenario Outline**

### **Scenario Outline:**

- uses Example keyword to define the test data for the Scenario
- **works for the whole test/scenario**
- Cucumber automatically run the complete test the number of times equal to the number of data in the Test Set

### **Data Table:**

- no keyword is used to define the test data
- **works only for the single step, below which it is defined**
- A separate code is needed to understand the test data and then it can be run single or multiple times but again just for the single step, not for the complete test

## **16. When in Cucumber some test scenarios fail and now you want to run failed ones, how would you do it?**

### **Modify existing runner class**

Add rerun:target/rerun.txt to your plugin

```
@RunWith(Cucumber.class)  
@CucumberOptions(features = { "." }  
    , glue = { "com/syntax/stepDefinitions" }  
    , plugin = {    "pretty",  
                  "html:target/cucumber-default-reports",  
                  "json:target/cucumber.json",  
                  "rerun:target/rerun.txt"}  
    , tags= {"@Temp"}  
    , monochrome = true  
    , dryRun = false)  
  
public class TestRunner {  
}
```

This will run all tests and then list all failed scenarios in rerun.txt file  
The file rerun.txt file is located in target folder.  
Rerun.txt contains path to the feature file and line number for scenario that failed

### Create another runner class

```
@RunWith(Cucumber.class)
@CucumberOptions(features="@target/rerun.txt"
    , glue = { "com/syntax/stepDefinitions" }
    , plugin = { "pretty",
        "rerun:target/rerun.txt",
        "html:target/cucumber-default-reports",
        "json:target/cucumber.json"}
    , monochrome = true
    , dryRun = false)

public class FailedRunner {

}
```

**17. Let's say I have Google app and in a Cucumber file I have 20 scenarios and there is 2 steps that are all common for all scenarios. How can I achieve that without repeating the code?**

Using background keyword

**18. In cucumber I have scenario with 10 steps and let's say step #5 I want to execute 5 times. How can I do it?**

Using DataTable