**Introduction to BDD-Cucumber**

**BDD – Behavioral Driven Development.**

**Approach to Develop,Test, maintain, Deliver the Software.**

**BDD is not a framework**

# **What is BDD?**

BDD is a way for software teams to work that closes the gap between **business people** and **technical people.**

BDD – 3 Amigos.

3 Amigos are – **Developers , Testers , Business Analyst/ Product Owner.**

Between these 3 Amigos , there is a constant Communication being conducted on daily basis.

Example –

**Redbus**

Encouraging collaboration across roles to build shared understanding of the problem to be solved

● Working in **rapid, small iterations** to increase **feedback** and the flow of value

● Producing system **documentation** that is automatically checked against the **system’s behaviour**

We do this by focusing collaborative work around concrete, real-world examples that illustrate how we want the system to behave. We use those examples to guide us from concept through to implementation, in a process of **continuous collaboration.**

# 

# 

# 

# **What is Cucumber?**



Cucumber reads executable **specifications** written in plain text and validates that the software does what those specifications say. The **specifications** consists of multiple *examples*, or *scenarios*.

**Cucumber is a framework that facilitates BDD, and brings BDD into Automation testing.**

There are 3 Main Components of Cucumber Framework :

1. **Feature files** - Files that are written in Plain English Format.

**2.** **Step Definitions -** Are the Subsequent Java classes that are used to convert Feature files into Java selenium Code.

**This acts like a translator which translates English language into java-selenium code.**

3. **Runner file-** Runner file takes the path of Feature file and Step Definition and Executes the code.

**Gherkin** - language/Format that is used to Write Feature files.

Cucumber reads executable specifications written in **plain text** and validates that the software does what those specifications say. The specifications consists of multiple *examples*, or ***scenarios***

**For example:**

**Scenario**: To validate Search box in my Youtube application.

**Given** I should open the chrome browser enter url (Pre conditions)

**When** I type shahrukhkhan in my search box

**And** I will press enter

**Then** All the songs related to srk should be displayed.

Scenario – Name of my scenario/ High level Description of scenario

Given – Preconditions/ Starting steps of test scenario

And – More Preconditions

When – Whatever the user is performing action

And – More actions from the user

Then – Expected Result.

And – multiple expected results

**Scenario**: To validate user is able to add to cart a mobile phone – iphone 14s

**Given** user should open chrome and enter flipkart url

**When** user searches for iphone14 s in searchbox

**And** user clicks on iphone 14s mobile

**And** user clicks on add to cart button

**Then** the product iphone 14 s should be displayed in Add to Cart page.

In order for Cucumber to understand the scenarios, they must follow some basic syntax rules, called [Gherkin](https://cucumber.io/docs/gherkin/).

**Installation of Cucumber :**

1. Go to Eclipse Marketplace, Help-> Eclipse Marketplace -> Search Cucumber and install Cucumber.

2. Add following Dependencies to Pom.xml

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>7.2.3</version>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-junit -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>7.2.3</version>

<scope>test</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.cucumber/cucumber-testng -->

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-testng</artifactId>

<version>7.14.0</version>

</dependency>

# **What is Gherkin?**

Gherkin is a set of grammar rules that makes plain text structured enough for Cucumber to understand. The scenario above is written in Gherkin.

Gherkin serves multiple purposes:

● Unambiguous executable specification

● Automated testing using Cucumber

● Document how the system *actually* behaves

Gherkin documents are stored in .feature text files and are typically versioned in source control alongside the software

# 

# **What are Step Definitions?**

[Step definitions](https://cucumber.io/docs/cucumber/step-definitions) connect Gherkin steps to programming code. A step definition carries out the action that should be performed by the step. So step definitions hard-wire the specification to the implementation.

**Day 8 :**

**BDD-Cucumber implementation with Junit.**

1. Setting up the Project:

● Create a new Maven project in your preferred IDE (Eclipse, IntelliJ, etc.).

● Add the necessary dependencies to your pom.xml file:

<dependencies>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>6.10.4</version> <!-- Use the latest version available -->

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>6.10.4</version> <!-- Use the same version as cucumber-java -->

<scope>test</scope>

</dependency>

</dependencies>

**2. Feature File:**

● Create a new directory named features in src/test/resources to store your feature files.

● Inside the features directory, create a feature file (e.g., sample.feature) with BDD scenarios:

Feature: Sample Feature

Scenario: Login with valid credentials

Given the user is on the login page

When the user enters valid username and password

Then the user should be logged in successfully

**3. Step Definitions:**

● Create a new package (e.g., stepdefinitions) in src/test/java to store your step definition classes.

● Inside the package, create a Java class (e.g., StepDefinitions) to define step definitions:

import io.cucumber.java.en.Given;

import io.cucumber.java.en.When;

import io.cucumber.java.en.Then;

public class StepDefinitions {

@Given("the user is on the login page")

public void navigateToLoginPage() {

// Implementation to navigate to the login page

}

@When("the user enters valid username and password")

public void enterValidCredentials() {

// Implementation to enter valid credentials

}

@Then("the user should be logged in successfully")

public void verifyLoginSuccess() {

// Implementation to verify successful login

}

}

**4. Test Runner:**

● Create a new Java class (e.g., TestRunner) in src/test/java to run your Cucumber tests:

import io.cucumber.junit.Cucumber;

import io.cucumber.junit.CucumberOptions;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@CucumberOptions(

features = "src/test/resources/features",

glue = "stepdefinitions"

)

public class TestRunner {

}

5. Running the Tests:

● Right-click on the TestRunner class and select "Run as JUnit Test" to execute your Cucumber tests.

● Cucumber will read the feature files, match the steps with step definitions, and execute the scenarios.

**BDD-Cucumber implementation with TestNG.**

BDD with Cucumber and TestNG combines the Cucumber framework, which uses Gherkin syntax for behavior-driven development, with TestNG for test execution in Java. The process involves writing feature files in Gherkin, implementing step definitions in Java using TestNG annotations, and executing the tests with a TestNG runner

1. Setting up the Project:

● Create a new Maven project in your preferred IDE (Eclipse, IntelliJ, etc.).

● **Add the necessary dependencies to your** pom.xml **file:**

**<dependencies>**

**<dependency>**

**<groupId>io.cucumber</groupId>**

**<artifactId>cucumber-java</artifactId>**

**<version>6.10.4</version> <!-- Use the latest version available -->**

**<scope>test</scope>**

**</dependency>**

**<dependency>**

**<groupId>io.cucumber</groupId>**

**<artifactId>cucumber-testng</artifactId>**

**<version>6.10.4</version> <!-- Use the same version as cucumber-java -->**

**<scope>test</scope>**

**</dependency>**

**</dependencies>**

2. Feature File:

● Create a new directory named features in src/test/resources to store your feature files.

● Inside the features directory, create a feature file (e.g., sample.feature) with BDD scenarios:

**Feature: Sample Feature**

Scenario: Login with valid credentials

Given the user is on the login page

When the user enters valid username and password

Then the user should be logged in successfully

**3. Step Definitions:**

● Create a new package (e.g., stepdefinitions) in src/test/java to store your step definition classes.

● Inside the package, create a Java class (e.g., StepDefinitions) to define step definitions:

**i**mport io.cucumber.java.en.Given;

import io.cucumber.java.en.When;

import io.cucumber.java.en.Then;

public class StepDefinitions {

@Given("the user is on the login page")

public void navigateToLoginPage() {

// Implementation to navigate to the login page

}

@When("the user enters valid username and password")

public void enterValidCredentials() {

// Implementation to enter valid credentials

}

@Then("the user should be logged in successfully")

public void verifyLoginSuccess() {

// Implementation to verify successful login

}

}

4. Test Runner:

● **Create a new Java class (e.g.,** TestRunner**) in** src/test/java **to run your Cucumber tests:**

**import io.cucumber.testng.AbstractTestNGCucumberTests;**

**import io.cucumber.testng.CucumberOptions;**

**@CucumberOptions(**

**features = "src/test/resources/features",**

**glue = "stepdefinitions"**

**)**

**public class TestRunner extends AbstractTestNGCucumberTests {**

**}**

5. Running the Tests:

● Right-click on the TestRunner **class and select "Run as TestNG Test" to execute your Cucumber tests.**

● **Cucumber will read the feature files, match the steps with step definitions, and execute the scenarios.**

**Programs :**

**package stepsdef;**

**import java.util.concurrent.TimeUnit;**

**import org.openqa.selenium.By;**

**import org.openqa.selenium.WebDriver;**

**import org.openqa.selenium.chrome.ChromeDriver;**

**import io.cucumber.java.en.Given;**

**import io.cucumber.java.en.Then;**

**import io.cucumber.java.en.When;**

**public class LoginSteps {**

**WebDriver driver;**

**@Given("user opens chrome and enters url")**

**public void user\_opens\_chrome\_and\_enters\_url() {**

**// Write code here that turns the phrase above into concrete actions**

**System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Downloads\\chromedriver-win64 (3)\\chromedriver-win64\\chromedriver.exe");**

**driver = new ChromeDriver();**

**driver.get("https://v1.training-support.net/selenium/login-form");**

**driver.manage().window().maximize();**

**driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);**

**}**

**@When("user enters username")**

**public void user\_enters\_username() {**

**driver.findElement(By.id("username")).sendKeys("Admin");**

**// Write code here that turns the phrase above into concrete actions**

**}**

**@When("user enters password")**

**public void user\_enters\_password() {**

**// Write code here that turns the phrase above into concrete actions**

**driver.findElement(By.id("password")).sendKeys("password");**

**}**

**@When("clicks on login")**

**public void clicks\_on\_login() {**

**// Write code here that turns the phrase above into concrete actions**

**driver.findElement(By.xpath("(//button[@type='submit'])[1]")).click();**

**}**

**@Then("Welcome back message is displayed")**

**public void welcome\_back\_message\_is\_displayed() throws InterruptedException {**

**// Write code here that turns the phrase above into concrete actions**

**System.out.println("login successful");**

**Thread.sleep(2000);**

**driver.close();**

**}**

**}**

**package runner\_;**

**import io.cucumber.testng.AbstractTestNGCucumberTests;**

**import io.cucumber.testng.CucumberOptions;**

**@CucumberOptions(**

**features = "C:\\Users\\Dell\\Documents\\RestAssuredWorkspace\\TestNgCucumber\_Form\\src\\test\\resources\\Login\\login.feature",**

**glue ="stepsdef"**

**)**

**public class loginRunner extends AbstractTestNGCucumberTests{**

**}**

**@tag**

**Feature: Login feature**

**@tag1**

**Scenario: validating login page**

**Given user opens chrome and enters url**

**When user enters username**

**And user enters password**

**And clicks on login**

**Then Welcome back message is displayed**

**Junit Runner file :**

**package runner;**

**import org.junit.runner.RunWith;**

**import io.cucumber.junit.Cucumber;**

**import io.cucumber.junit.CucumberOptions;**

**@RunWith(Cucumber.class)**

**@CucumberOptions(**

**features = {"C:\\Users\\Dell\\Documents\\RestAssuredWorkspace\\SimpleFormProject\_Cucumber\\src\\test\\resources\\Forms\\form.feature"},**

**glue = {"steps"}**

**)**

**public class FornRunner {**

**}**

**package steps;**

**import java.util.concurrent.TimeUnit;**

**import org.openqa.selenium.Alert;**

**import org.openqa.selenium.By;**

**import org.openqa.selenium.WebDriver;**

**import org.openqa.selenium.chrome.ChromeDriver;**

**import io.cucumber.java.en.Given;**

**import io.cucumber.java.en.Then;**

**import io.cucumber.java.en.When;**

**public class FormSteps {**

**WebDriver driver;**

**@Given("user opens chrome browser and enters simpleform url")**

**public void user\_opens\_chrome\_browser\_and\_enters\_simpleform\_url() {**

**// Write code here that turns the phrase above into concrete actions**

**System.setProperty("webdriver.chrome.driver", "C:\\Users\\Dell\\Downloads\\chromedriver-win64 (3)\\chromedriver-win64\\chromedriver.exe");**

**driver = new ChromeDriver();**

**driver.get("https://v1.training-support.net/selenium/simple-form?");**

**driver.manage().window().maximize();**

**driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);**

**}**

**@Given("textbox should be enabled")**

**public void textbox\_should\_be\_enabled() {**

**driver.findElement(By.id("firstName")).isEnabled();**

**}**

**@When("user enters text in firstname, lastname, email,conatct, message")**

**public void user\_enters\_text\_in\_firstname\_lastname\_email\_conatct\_message() {**

**driver.findElement(By.id("firstName")).sendKeys("Akshay");**

**driver.findElement(By.id("lastName")).sendKeys("Koulgi");**

**driver.findElement(By.id("email")).sendKeys("Koulgi@gmail.com");**

**driver.findElement(By.id("number")).sendKeys("9090923851");**

**driver.findElement(By.xpath("//textarea")).sendKeys("Hi my name is Akshay and I am your selenium Trainer");**

**}**

**@When("user clicks on sumbit button")**

**public void user\_clicks\_on\_sumbit\_button() {**

**// Write code here that turns the phrase above into concrete actions**

**driver.findElement(By.xpath("//input[@value='submit']")).click();**

**}**

**@Then("An alert should be displayed")**

**public void an\_alert\_should\_be\_displayed() {**

**System.out.println("ALert is present");**

**Alert alert = driver.switchTo().alert();**

**alert.accept();**

**driver.close();**

**// Write code here that turns the phrase above into concrete actions**

**}**

**}**

**@tag**

**Feature: Form Feature**

**@smoke**

**Scenario: To validate simple form and all the textboxes present**

**Given user opens chrome browser and enters simpleform url**

**And textbox should be enabled**

**When user enters text in firstname, lastname, email,conatct, message**

**And user clicks on sumbit button**

**Then An alert should be displayed**