Cucumber Introduction

Cucumber is a **testing tool** used in **Behavior-Driven Development (BDD)** that allows developers, testers, and business analysts to **write test cases in plain English** using a language called **Gherkin**.

🥒 What is Cucumber?

Cucumber is an **open-source automation tool** for writing tests that are easy to read and understand, even for non-programmers.
It bridges the gap between **technical** and **non-technical** team members by using a **simple, readable format** for test cases.

👥 Who Uses Cucumber?

- **Business Analysts** write the requirements in plain English.
- **Testers** automate the test steps using tools like Selenium.
- **Developers** connect the plain English steps to actual code logic.

🗣️ What is Gherkin?

Gherkin is the **language** used by Cucumber to define test cases. It uses specific keywords like:

- `Feature`: A feature to be tested
- `Scenario`: A test case
- `Given`: Pre-condition
- `When`: Action performed
- `Then`: Expected result
- `And` / `But`: Additional steps

✅ Sample Gherkin Scenario:

```gherkin
Feature: Login Functionality

  Scenario: Valid Login
    Given User is on the login page
    When User enters valid username and password
    And clicks on the login button
    Then User should be redirected to the dashboard
```

🧪 How It Works (Behind the Scenes)

1. **Write Gherkin scenarios** in a `.feature` file.
2. **Create Step Definitions** (code behind each step in Java, Python, etc.).
3. **Run tests** with a test runner (like JUnit/TestNG).
4. **Cucumber executes the steps**, matching them with the actual code.

🔗 Common Cucumber Integrations:

- **Selenium** – for browser automation
- **JUnit/TestNG** – for running tests
- **Maven/Gradle** – for managing dependencies
- **CI/CD tools** – Jenkins, GitHub Actions

🧠 Benefits of Using Cucumber

- Easy to understand for non-programmers
- Encourages collaboration (Dev + QA + BA)
- Supports many languages (Java, Python, Ruby, etc.)
- Helps maintain up-to-date documentation

Would you like to try a **simple hands-on example** with Cucumber and Selenium in Java?

Great question! Let's break it down simply 👇

✅ Why Use Cucumber in Testing? (For Beginners)

Cucumber is used in testing to make **communication easy**, **testing clearer**, and **collaboration smoother** — especially between **developers**, **testers**, and **business people**.

🌟 1. Plain English Test Cases

Cucumber uses **Gherkin**, a language that reads like plain English.

🧾 Example:

```
gherkin

Scenario: Successful login
  Given the user is on the login page
  When the user enters valid credentials
  Then they should see the dashboard
```

✅ Easy to understand for **non-technical people**.

🤝 2. Better Collaboration (BA + Dev + QA)

- Business Analysts write **requirements**.
- Testers write **automated test scripts** based on those.
- Developers understand and use the same language.

Everyone works **on the same page**.

🧪 3. Automated Testing

Cucumber integrates with tools like:

- **Selenium** for browser testing
- **JUnit/TestNG** for test execution
- You write Gherkin steps → then link to **real code** that runs the tests.

📋 4. Living Documentation

Your feature files become **live documentation** of how the system should behave.
They **stay updated** because they're directly tied to the test code.

🔁 5. Reusability and Clean Code

- You can **reuse steps** like **`Given the user is logged in`**.

- Keeps code **modular** and **clean**.

🚀 6. Supports Agile and BDD

Cucumber fits perfectly into **Agile** and **Behavior-Driven Development (BDD)**:

- Focus on **user behavior**
- Develop **tests before writing code** (Test-First)
- Catch bugs **early**

🧠 Summary: Why Use Cucumber?

Reason	Benefit
✅ Readable tests	Written in plain English
👥 Teamwork	BA + QA + Dev collaborate
🔁 Reusable steps	Write less, test more
📜 Documentation	Always up-to-date
🚀 BDD support	Test real user behavior

Would you like a beginner-friendly **Cucumber + Selenium test project setup** next?

Great question! Understanding the difference between **Cucumber** and **Selenium** is key when you're starting out in test automation.

Let's compare them in a beginner-friendly way 👇

🥒 Cucumber vs 🕷️ Selenium (For Beginners)

Feature	Cucumber	Selenium
What is it?	A **BDD tool** used to write test scenarios in **plain English (Gherkin)**	A **browser automation tool** used to interact with web applications
Purpose	Describes **what** should be tested	Performs the actual **automation** and interaction with web pages
Language	Uses **Gherkin syntax** (Given, When, Then)	Uses programming languages like **Java, Python, C#**
Technical Skill?	Minimal for writing test scenarios	Requires **coding skills** to write automation scripts
Who uses it?	BAs, QAs, Testers, Developers (all can understand)	Mostly **Testers and Developers**
Can it run tests?	No, it defines **what to test**	Yes, it can **automate and run** tests on browsers
Integration	Needs Selenium or another tool to actually run UI tests	Can be used **alone** or **with Cucumber** for BDD
Example	**`Given the user is on the login page`**	**`driver.get("https://example.com/login");`**

🧠 Easy Analogy:

Role	Example
Cucumber	Like writing a movie script – easy to read and understand.
Selenium	Like the camera and crew actually shooting the movie based on the script.

✅ Use Cucumber when:

- You want **plain English** test cases (BDD).
- You want **team collaboration** (BA + QA + Dev).
- You want **living documentation** of your system behavior.

✅ Use Selenium when:

- You need to **interact with the browser**.
- You're writing detailed test scripts for **clicks, inputs, alerts, etc.**.
- You're performing **functional UI automation**.

🧩 Use Them Together:

- **Cucumber** writes the test case in plain English.
- **Selenium** performs the actual test steps.

👉 Example:

```gherkin
Given the user is on the login page
When the user enters valid credentials
Then the user should see the dashboard
```

⬇️ (Linked in code to Selenium like)

```java
driver.get("https://example.com/login");
driver.findElement(By.id("username")).sendKeys("admin");
driver.findElement(By.id("password")).sendKeys("admin123");
driver.findElement(By.id("loginBtn")).click();
```

Would you like a **hands-on mini project** showing how to use **Cucumber + Selenium** together?

Absolutely! Let's explore how **Cucumber is used in real-time projects** — with examples and explanations that are beginner-friendly 👇

🏗️ Cucumber in Real-Time Projects

In real-world projects, Cucumber is used as part of **Behavior-Driven Development (BDD)** to make testing **collaborative, readable, and maintainable**.

📌 Where Cucumber Fits in the Real World

💼 Typical Setup in a Real Project:

Layer	Tools Used
🧾 Requirements	Written by Business Analyst
🧪 Test Cases	Written in **Cucumber (Gherkin)**
🖥️ Automation	Performed using **Selenium/WebDriver**
🧑‍💻 Language	Java/Python (for step definitions)
🚀 CI/CD	Jenkins/GitHub Actions/Bitbucket Pipelines
📦 Build Tool	Maven or Gradle
🔬 Reports	Cucumber HTML Reports, Extent Reports

🧩 Example Real-Time Scenario: Login Feature

🧾 1. Feature File (`Login.feature`)

```gherkin
Feature: Login Functionality

  Scenario: Valid Login
    Given the user is on the login page
    When the user enters username "admin" and password "admin123"
    And clicks the login button
    Then the user should be redirected to the dashboard
```

💻 2. Step Definition (Java + Selenium)

```java
@Given("the user is on the login page")
public void user_on_login_page() {
    driver.get("https://example.com/login");
}

@When("the user enters username {string} and password {string}")
public void user_enters_credentials(String username, String password) {
    driver.findElement(By.id("username")).sendKeys(username);
    driver.findElement(By.id("password")).sendKeys(password);
}

@And("clicks the login button")
```

```
public void user_clicks_login() {
    driver.findElement(By.id("loginBtn")).click();
}

@Then("the user should be redirected to the dashboard")
public void user_redirected_to_dashboard() {
    Assert.assertTrue(driver.getCurrentUrl().contains("dashboard"));
}
```

🧪 3. Execution

- You run your tests using **JUnit/TestNG**.
- Cucumber reads `.feature` files.
- Matches steps with Java code (step definitions).
- Selenium automates the browser.

✅ Benefits in Real-Time Projects

Benefit	Explanation
👫 Collaboration	BAs write feature files, QAs automate them, Devs support integration.
📖 Clear Test Cases	Everyone can read and review test scenarios easily.
🧪 Reusable Steps	Steps like “login to the app” are reused across multiple tests.
📊 Reporting	Easy to generate **HTML reports** for test runs.
🔁 CI Integration	Cucumber tests run automatically after each code push (CI/CD).

💡 Common Use Cases in Projects

1. **Login/Logout**
2. **User Registration**
3. **Search and Filters**
4. **Shopping Cart**
5. **Payment Gateway Validation**
6. **Error/Validation Messages**
7. **API Testing (with Cucumber + Rest Assured)**

🛠️ Sample Folder Structure in Real Projects

```
bash

src
├── test
│   ├── java
│   │   ├── stepdefinitions
│   │   ├── hooks
│   │   └── runners
│   └── resources
│
```

```
│         └── features
│             └── Login.feature
```

Would you like a **complete real-time mini project** with Cucumber + Selenium + Maven setup? I can guide you step-by-step.

Perfect! Here's a **complete real-time mini project** using **Cucumber + Selenium + Maven** — beginner-friendly and suitable for hands-on learning.

🧪 Mini Project: Automate OrangeHRM Login Using Cucumber + Selenium

---

🔧 Step 1: Project Setup (Maven + Java)

📁 Folder Structure

```css
CucumberOrangeHRM/
├── src/
│   ├── main/
│   └── test/
│       ├── java/
│       │   ├── stepdefinitions/
│       │   ├── runners/
│       │   └── pages/
│       └── resources/
│           └── features/
└── pom.xml
```

📦 Step 2: `pom.xml` – Add Dependencies

```xml
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.orangehrm</groupId>
  <artifactId>CucumberOrangeHRM</artifactId>
  <version>1.0</version>

  <dependencies>
    <!-- Selenium -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>4.21.0</version>
    </dependency>

    <!-- Cucumber -->
    <dependency>
      <groupId>io.cucumber</groupId>
      <artifactId>cucumber-java</artifactId>
      <version>7.14.0</version>
    </dependency>
    <dependency>
      <groupId>io.cucumber</groupId>
      <artifactId>cucumber-junit</artifactId>
      <version>7.14.0</version>
    </dependency>

    <!-- JUnit -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.13.2</version>
    </dependency>
    <!-- WebDriverManager -->
    <dependency>
      <groupId>io.github.bonigarcia</groupId>
      <artifactId>webdrivermanager</artifactId>
      <version>5.8.0</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>3.1.2</version>
      </plugin>
    </plugins>
  </build>
</project>
```

📄 Step 3: `Login.feature` – Feature File

📁 **`src/test/resources/features/Login.feature`**

```
gherkin

Feature: OrangeHRM Login

  Scenario: Login with valid credentials
    Given user is on login page
    When user enters username "Admin" and password "admin123"
    And clicks on login button
    Then user should land on dashboard page
```

💡 Step 4: Step Definitions

📁 **`src/test/java/stepdefinitions/LoginSteps.java`**

```
java

package stepdefinitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import io.github.bonigarcia.wdm.WebDriverManager;
import io.cucumber.java.en.*;
import org.junit.Assert;

public class LoginSteps {
    WebDriver driver;

    @Given("user is on login page")
    public void user_is_on_login_page() {
        WebDriverManager.chromedriver().setup();
        driver = new ChromeDriver();
        driver.get("https://opensource-demo.orangehrmlive.com/");
        driver.manage().window().maximize();
    }

    @When("user enters username {string} and password {string}")
    public void user_enters_username_and_password(String username, String password) {
        driver.findElement(By.name("username")).sendKeys(username);
        driver.findElement(By.name("password")).sendKeys(password);
    }
    @And("clicks on login button")
    public void clicks_on_login_button() {
        driver.findElement(By.tagName("button")).click();
    }

    @Then("user should land on dashboard page")
    public void user_should_land_on_dashboard_page() throws InterruptedException {
        Thread.sleep(2000); // wait for page to load
        boolean status = driver.getCurrentUrl().contains("dashboard");
        Assert.assertTrue(status);
        driver.quit();
    }
}
```

▶️ Step 5: Test Runner

📁 **`src/test/java/runners/TestRunner.java`**

```
java

package runners;

import org.junit.runner.RunWith;
import io.cucumber.junit.Cucumber;
import io.cucumber.junit.CucumberOptions;

@RunWith(Cucumber.class)
@CucumberOptions(
  features = "src/test/resources/features",
  glue = "stepdefinitions",
  plugin = {"pretty", "html:target/cucumber-reports"},
  monochrome = true
)
public class TestRunner {
}
```

✅ Step 6: Run the Test

- Open terminal or use your IDE.
- Run the test with:

```
bash

mvn test
```

🔍 It will:

- Launch the browser
- Go to the OrangeHRM login page
- Enter credentials
- Click login
- Validate successful login to dashboard

📊 Output

- **Browser automation** will run using Selenium.
- **Gherkin steps** will be logged in the console.
- **HTML report** will be generated in `target/cucumber-reports/index.html`.

🎉 Congrats!

You’ve just created a real-time working **Cucumber + Selenium mini project** using:

- Java
- Maven
- Gherkin feature file
- Selenium WebDriver
- JUnit test runner

Would you like to **add Page Object Model (POM)** to this setup next for better code structure?

Here are the **key advantages of using Cucumber for QA (Quality Assurance)** — explained in a beginner-friendly way 👇

✅ Advantages of Cucumber for QA Testers

---

1. 🗣️ Readable Test Cases (Plain English)

- Cucumber uses **Gherkin**, which reads like **natural language**.
- Test scenarios are **easy to write, read, and understand** — even for non-technical team members.

🧾 Example:

```
gherkin

Scenario: Search a product
  Given user is on homepage
  When user searches for "laptop"
  Then results related to "laptop" should be displayed
```

✅ This allows QAs to **focus on behavior**, not just code.

---

2. 🤝 Improved Collaboration (BA + QA + Dev)

- Cucumber supports **Behavior-Driven Development (BDD)**.
- Everyone — Business Analysts, Developers, Testers — can **collaborate using the same language**.
- Reduces misunderstandings between QA and dev teams.

---

3. 📋 Acts as Live Documentation

- Feature files **double as documentation**.
- Easy to keep up-to-date because they're linked directly to test code.
- No need for extra Excel or Word test cases — your `.feature` files *are* the documentation.

---

4. ♻️ Reusable Step Definitions

- Common actions like `login`, `add to cart`, etc. can be **reused** across different test scenarios.

🔁 Reduces **duplicate code** and improves **maintainability**.

---

5. 🧪 Easy Integration with Selenium & API Tools

- Cucumber integrates easily with:
  - **Selenium** for UI testing
  - **Rest Assured** for API testing

- **JUnit/TestNG** for test runners
- **Jenkins/GitHub Actions** for CI/CD

🧩 QA can automate **end-to-end**, UI, or API tests using the same framework.

---

6. 🚀 CI/CD Friendly

- You can run Cucumber tests automatically in a CI pipeline (e.g., Jenkins, GitLab, GitHub Actions).
- Helps QAs catch bugs **early and often**.

---

7. 📊 Reports and Traceability

- Supports multiple **reporting formats** (HTML, JSON, ExtentReports).
- Easily trace which feature/scenario passed or failed.

---

8. 🧠 Test-First Mindset (BDD)

- Encourages QAs to **think from a user's perspective**.
- Write scenarios **before the application is built**.
- Results in **better test coverage** and **fewer missed edge cases**.

🧠 Summary: Why QAs Love Cucumber

Advantage	Benefit to QA
✅ Readable	Easy to write and explain
🤝 Collaborative	Aligns with BA & Dev
♻️ Reusable	Saves time and effort
📄 Self-documenting	No need for separate test docs
🔧 Integratable	Works with Selenium, API tools
🚀 CI/CD Ready	Fits into modern DevOps
🧠 Behavior-focused	Ensures real user experience is tested

Would you like a **cheat sheet or template** to help write good Gherkin scenarios for QA testing?