

Gherkin: Suggested Best Practices

Sheffer, Jared

Table of Contents

[Gherkin: Suggested Best Practices 2](#_Toc480972153)

[Keep track of what scenarios included in a requirement 2](#_Toc480972154)

[Combine Your Common Scenarios 3](#_Toc480972155)

[Make your steps modular 4](#_Toc480972156)

[5 Rules of Modularity 5](#_Toc480972157)

[Rule 1: If multiple steps do the same thing make it obvious 5](#_Toc480972158)

[Rule 2: Keep case consistent: lowercase recommended 5](#_Toc480972159)

[Rule 3: Keep tables consistent 6](#_Toc480972160)

[Rule 4: Include all needed information as if the feature, scenario, or step were alone 6](#_Toc480972161)

[Rule 5: Don’t use Microsoft Word to create your files 7](#_Toc480972162)

# **Gherkin: Suggested Best Practices**

## **Keep track of what scenarios included in a requirement**

* 1. It allows you to easily trace a scenario to a requirement.
  2. This helps to verify that you have complete coverage for that requirement even if it changes
  3. If you keep a common format for the connection than you can code a formatter as part of your test execution to show for example how many of a particular requirement ran or failed.
  4. If it is included in the scenario description than it will be visible in the test report if it passes or fails.

Example:

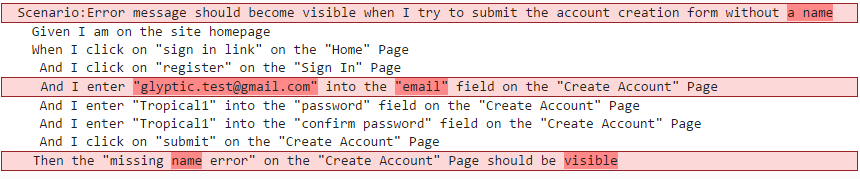
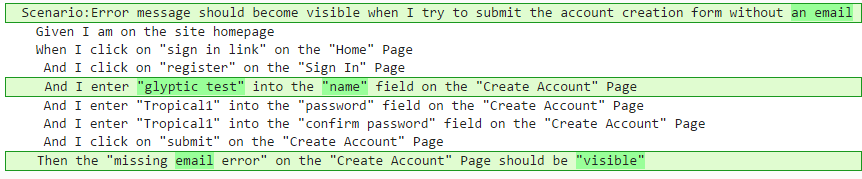
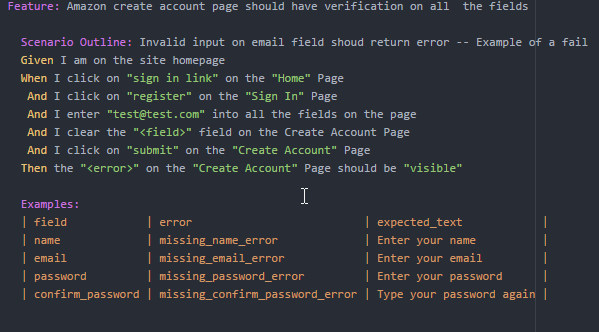
Feature: If the whole feature covers requirements: RLS2094, RLS2031, RLS4756

Scenario: Description of the scenario followed by the requirement: RLS:1003

## **Combine Your Common Scenarios**

* 1. If most steps are the same
  2. Consider a scenario outline combination
  3. This helps promote maintainable scripts since if you must change the flow than you only need modify it in one place vs multiple places

Example:



## **Make your steps modular**

* 1. Be constant with your language.
  2. Use parameters and make them clear
  3. Make sure that if your steps do the same thing that they are worded the exact same way
  4. Follow the 5 rules of modulatory
  5. Modularity will help make automation of these scripts easier because it will allow the use of modern frameworks and eliminate code duplication
  6. It will be easier to read and create the gherkin

1. It will help all the projects get on the same page

# **5 Rules of Modularity**

## **Rule 1: If multiple steps do the same thing make it obvious**

* When determining if steps are different or not you should ignore the keywords (i.e. Given When Then…) Cucumber ignores these when matching a step definition. Just because one has a Given and one has a When it does not make them 2 different behaviors and the step definition would be a common single step definition if done properly
* Add quotes to make if obvious if something can be swapped out.

**Example:**

**Given I click on Sign In link on the Home page**

**When I click on the Register Button on the Sign In page**

**When I click on “something” on the “somewhere” page**

## **Rule 2: Keep case consistent: lowercase recommended**

* Cucumber uses regular expressions to match your steps. If you are not consistent with your wording than either. The regular expression must be updated to take all your possible wordings into account
  + Which leads to code duplication
  + Which leads to unmaintainable tests
* Or the developer must update your steps to make them consistent.

**Example:**

**Given I Click on “Sign In link” on the “Home” page**

**Given I click on “Sign In link” on the “Home” page**

These will not execute the same step

## **Rule 3: Keep tables consistent**

* When developing an application or an automation test suite it is common practice to down case all data in which case is not a factor.
* It is suggested to keep your variables and headers all lowercase and underscore separated

**Example:**

**Then the "<Error>" should be "visible" and contain "<Expected Text>"**

**Vs**

**Then the "<error>" should be "visible" and contain "<expected\_text>"**

**Examples:**

**| error | expected\_text |**

## **Rule 4: Include all needed information as if the feature, scenario, or step were alone**

* Features and scenarios should never rely on other features or scenarios
* Each feature, scenario, and step should be independent.
* They should not require data to be passed between them
* Steps can rely on other steps but should not share data

**Example:**

**When I click on "sign in "**

**And I enter the "user0 " data**

**vs**

**When I click on "sign in link" on the "home" page**

**And I enter the "user0" data from the "dev\_users.json" file**

## **Rule 5: Don’t use Microsoft Word to create your files**

* **Microsoft uses smart quotes so don’t use it to write your gherkin. If you do then cleanse it before delivery**
* **The smart quotes are not rendered in the same manner as the standard quote character**

**Example:**

**When I click on "sign in link" on the "home" page**

**When I click on “sign in link” on the “home” page**

**- Microsoft uses smart quotes**