

**All the Steps for STLC**

**Customer Requirements Analysis**

Analyse what does the customer need to get at the end of the project.

* Usually written in Backlog file

**Test Planning**

Plan how my team and I will get to test to certify all the customer requirements will get fullfilled at the end of the DLC.

* Security;
* Loading time;
* Stress;
* Design;
* Usability;
* Intuitiveness.

**Test Cases Design and Development**

Design the step-by-step process of each test case and develop it.

E.g.:

1. User Story: **As an** user, **I want to** buy an arcticle from this website, **so that** I will go until the sales page and buy it.
2. Test cases:
   1. **TC01 Happy path** - I do it the conventional way and it works.
   2. **TC02 Wrong path** - I do it the wrong way and it doesn’t work.
   3. **TC03 White hack** - I try to break any settled rule.
3. Setup the expected behavior of the test cases:
   1. **TC01** - Should work.
   2. **TC02** - Shouldn’t work.
   3. **TC03** - I shouldn’t find any gap to hack it.

**Test Environment Setup**

Setup what will be the version of the tested software, the frameworks, tools, languages and how it will be presented.

E.g.:

* Software Version 1.0
* Web plataform
* Windows 10
* Selenium-Webdriver
* JavaScript
* Jira and Confluence

**Test Execution**

Execute the test and catch all the behaviors/results.

* Note all the actual behavior of the test cases:
  + **TC01** - It worked/passed.
  + **TC02** - Shouldn’t work and it didn’t. (Passed.)
  + **TC03** - I didn’t find any gap to hack it. (Passed.)

* **Bug Report:**

Once one of the TC’s didn’t pass, (expected behavior isn’t the same as the actual behavior) I must to report the bug.

* + Tools: Jira, Bugzilla, Mantis; It’s also commonly reported in Excel and WordExpress.
  + Crucial information to contain:
    1. Software Version;
    2. Specific Browser;
    3. Operating System;
    4. Branch;
    5. Step-by-step to reproduce it;
    6. Severity;
    7. Priority.

**Test Closure**

Analyse all the results and make the test report.

* Usually made in BDD (Behavior-Driven Development) that is a natural way to express the **behavior** of the tested software.
* BDD most common tool: **Cucumber**.
* Cucumber language: **Gherkin**.
  + How it is commonly made:
    - Type this code in integrated terminal: npm install --sava-dev @cucumber/cucumber
  + How it is written:
    - * **Feature:** Sales page
      * **Scenario:** I’m buying an arcticle
      * **Given** I’m on the website’s sales page “https://www.exemple/david/salespage.com”
      * **And** the internet connection is on
      * **When** I click on the arcticle
      * **And** the arcticle is visible in the cart
      * **And** I check in my details
      * **And** I pay for it
      * **Then** the “Thank you” page is trigged “https://www.exemple/thankpage.com”
      * **And** the arcticle I just bought is on to-be-shipped page “https://www.exemple/tobeshipped.com”

By J. David S. Vilas Boas ©