

# Codeception - Performance Problem

# Acceptant Tests

Acceptant Tests are performed to verify whether end-users requirements are fulfilled for acceptability .

1. we write the code ( with Codeception ) to produce the test's steps
2. these steps execute the same action that a User do to complete the test directly on the browser
3. at the end we receive a report that describe the test's result .

## ADD USER

Username \*



Full Name \*



Password \*



Confirm Password \*



Email



Phone number



Note



Timezone



Role












SAVE

CANCEL

```
83
84
85     $I->click( link: 'New User');
86     $I->waitForElement( element: '#userForm', timeout: 5);
87     $I->seeCurrentUrlEquals( uri: '/igacadmin/user/edit');
88     // (3)
89     $I->fillField( field: 'username', $this->UserToCreate);
90     $I->fillField( field: 'fullname', value: 'test user');
91     $I->fillField( field: 'password', $this->PasswdToCreate);
92     $I->fillField( field: 'confirm_password', $this->PasswdToCreate);
93     $I->fillField( field: 'email', value: $this->UserToCreate.'@user.it');
94     $I->fillField( field: 'phone', value: '+393333333333333');
95     $I->fillField( field: 'notes', value: 'Some note');
96
97     $I->click( link: 'save');
98     $I->waitForElement( element: '.resultMessage', timeout: 5);
99     $I->see( text: "User '$this->UserToCreate' inserted/modified successfully");
100
101
102
```

## ADD USER

Username *	<input type="text" value="Utente01"/>	
Full Name *	<input type="text" value="prova01"/>	
Password *	<input type="password" value="....."/>	
Confirm Password *	<input type="password" value="....."/>	
Email	<input type="text" value="utente@gmail.com"/>	
Phone number	<input type="text" value="333229845"/>	
Note	<input type="text" value="testing"/>	
Timezone	<input type="text" value="UTC"/>	
Role	<input type="text" value="-"/>	

SAVE

CANCEL

# Codeception Results **OK** (58.8s)

## Acceptance Tests

+ CustomerManagerSuiteCest » Set env 3.78s

+ CustomerManagerSuiteCest » Create a New Customer 8.95s

+ CustomerManagerSuiteCest » Create a New Customer but don't Save it 9.32s

+ CustomerManagerSuiteCest » Create a New Customer but already exists 8.79s

- CustomerManagerSuiteCest » Update a Customer 12.9s

+ I login "admin","admin","Default Customer","Default SubDomain"

+ I create new customer "Customer-03","Customer-03@netresults.it"

+ I click edit or delete "Customer-03","Edit"

I check next page js "//h2[contains(text(),'Edit Customer')]"

+ I insert and check customer form "Customer-04","Customer-04@netresults.it","VALID\_UPDATE"

+ I delete customer from list "Customer-04"

+ CustomerManagerSuiteCest » Delete a Customer but no confirm 5.94s

+ CustomerManagerSuiteCest » Delete a Customer 6.04s

+ CustomerManagerSuiteCest » Reset env 3.09s

## Summary

Successful scenarios: **8**

Failed scenarios: **0**

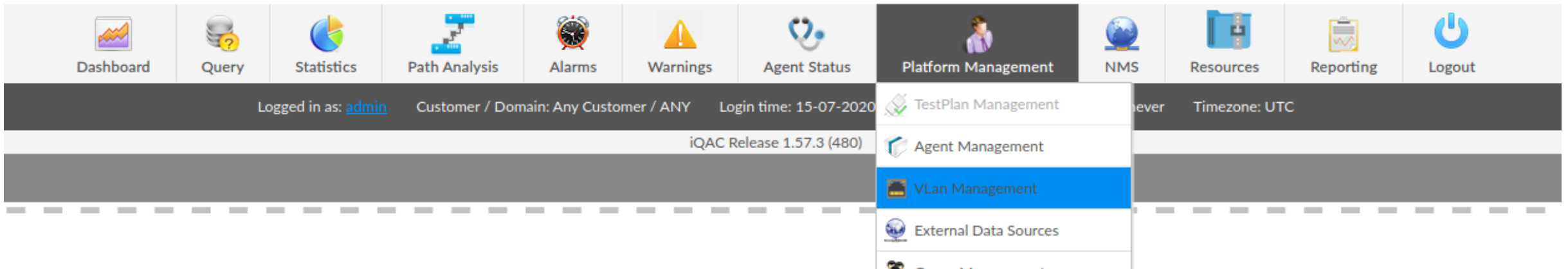
Skipped scenarios: **0**

Incomplete scenarios: **0**

# Example Performance Problem

We are in the Homepage ( index page ) and we are going to do the following steps :

1. would like to click on the "Vlan Management " link on the Navbar
- 2.
3. Go to the "Vlan Management" page and recognise the title "<h2> Vlan Management</h2>"



# Expected Result

Dashboard Query Statistics Path Analysis Alarms Warnings Agent Status Platform Management NMS Resources Reporting Logout

Logged in as: [atena](#) Customer / Domain: ATENA-TEST / NR-test Login time: 21-07-2020 10:03:07 Session expiration time: never Timezone: Europe/Rome

## VLAN MANAGEMENT

Name	Vlan tag	Address	Gateway	Mask	MTU
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

◀ ◀ 0/0 ▶ ▶ 10 ▼

[Get sample provisioning file](#) **IMPORT FROM CSV** **EXPORT TO CSV**

Expected element to be found ( with \$I->waitForElement )

- `<h2> VLAN MANAGEMENT </h2>`



## Vlan Management Link Element Structure

- Html Code – Vlan Management <a> link ( button to click )

```
<a  
  href="/atenadmin/custom_vlan/index"  
  id="vlan_management"  
>  
Vlan Management</a>
```



- Codeception Code

```
$I->SeeCurrentUrlEquals('/atena/dashboard/index');  
$I->moveMouseOver('#admin');  
  
$I->waitForElement(element: '//a[@href="/atenadmin/custom_vlan/index"][@id="vlan_management"]');  
$I->click(link: '#valn_management')  
  
$I->waitForElement(element: '//h2[contains(text(), Vlan Management)]', timeout:5);
```

## TEST RESULT

\$I->SeeCurrentUrlEquals('/atena/dashboard/index');	• OK
\$I->moveMouseOver('#admin');	OK
\$I>waitForElement('//a[@href="/atenadmin/custom_vlan/index"] [@id="vlan_management"]');	OK
\$I->click('#vlan_management');	OK
\$I->waitForElement('//h2[contains(text(),Vlan Management)]', 5);	FAIL

### Failed Test

After clicking the button "vlan management"

- the browser remain in the "Homepage" ( index page )
- Codeception didn't found the element "<h2>Vlan Management</h2>" ( that corripsond to the title in the "Vlan Management"Page
-

## What has happened in the page "Homepage"

- 

### Codeception's Actions :

1. It recognise the "`<a> link element </a>`" and the **associated link**
- 2.
3. click , correctly , the "`<a> link element </a>` "
- 4.
- 5.

### Browser Reaction :

1. The "`<a> link element </a>`", with the correct link associated , was loaded ( from the DOM side )
- 2.
3. Once the "`<a> link element </a>`" was clicked , this element doesn't respond correctly because it wasn't active yet .

# Performance Problem

## Introduction

From the previous example we understand that :

- even if the document's elements are correctly loaded and recognised by Codeception checks ,
- these elements are not still ready to be used by the User .

For this reason , we need to wait that **all the page's process are completely loaded** before take any action on the page .

# Loading Page process

As we have seen in the example before , we could simplify the loading page process in two steps:

- 1. HTML elements Loaded:** When the page load all the page elements and images ( that is similar at the document.ready == complete )
  - In the Vlan Example this step correspond when the "`<a>Vlan_Management<a>`" element is loaded
  - But the Elements status are "not active"
- 2.
- 3. Window process :** when the page will load all the process that give the "actions" ( like clickable etc.. ) to the page's element
  - In the Vlan Example this step correspond when you click on the "`<a>Vlan_Management<a>`" element and it will lead you to the "Vlan Management" Page
  - The elements status are "active"

To **fully load a page** you need to wait the "HTML elements loaded" and the "Window process" .

When a page is fully loaded , it will responde correctly to all the action taken in

# Performance Problem

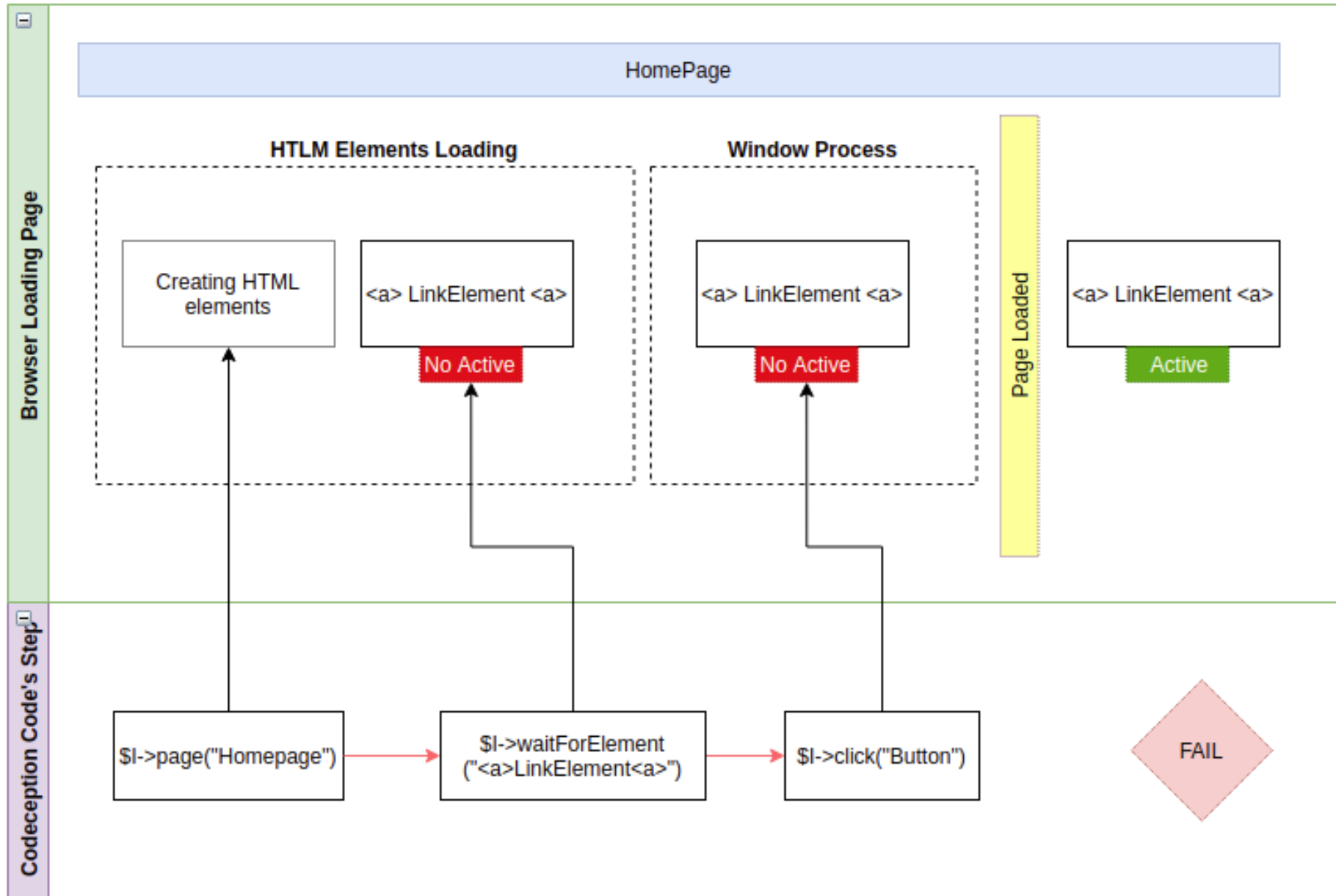
The test's performance depends by the Environment's attribute like :

- Computer structure ( RAM , CPU , etc .. )
- Network configuration ( internet , )
- Which Virtual Machine are you using , etc..

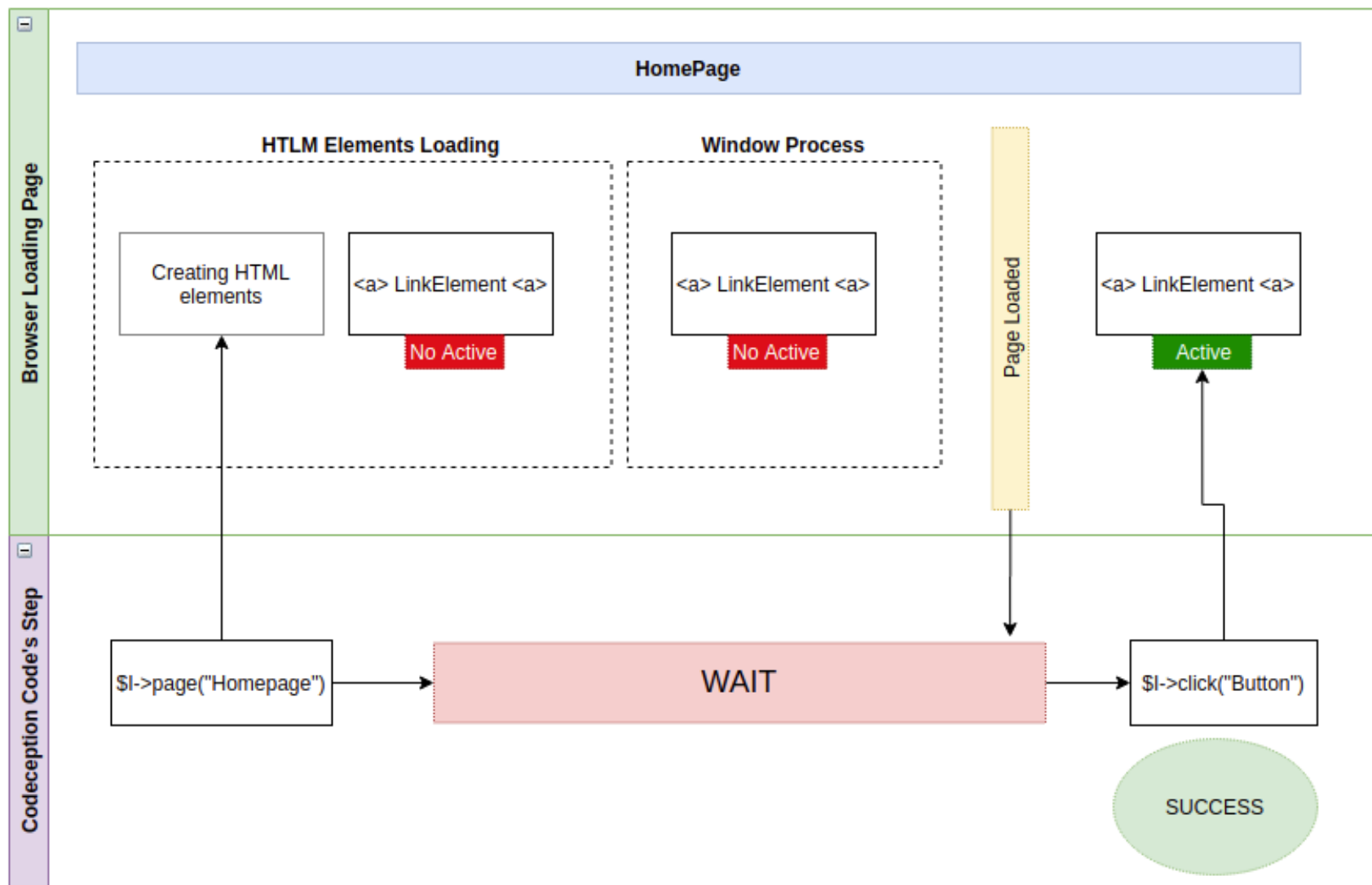
Launching tests with different environments it will modify the **Codeception's action speed** and the **browser loading process** .

**Important** : If the codeception's code steps are not synchronized with the browser's loading process , the test will fail

# Click Vlan Management : Error



# Click Vlan Manadement : Success with



We need a function that let Wait ( stop ) Codeception's code untill the page is correctly loaded ( with all the elements active ) or better to synchronise the codeception's code to the browser load process



# How to manage Codeception's "WAIT"

I considered two type of wait : "**static wait**" and "**relative wait**" .

1. "**static wait**" is the classic function that will impose to wait for a definite and constant time like "10 seconds"

- Example : " \$I->wait(10) ; "
- You'll stop the codeceptoin for 10 seconds and , after that , the next Codeception's action will start

2.

3. "**relative wait**" : they are function that will stop the code ( for a maximum \$timeout amount of time ) untill you see a determinate page's element or a situation become true .

- Example : " \$I->waitForElement("linkElement , 5") ; " or "\$I->waitForJs(10)"
- You'll stop the codeceptoin for 10 seconds and , after that , the next Codeception's action will start

•

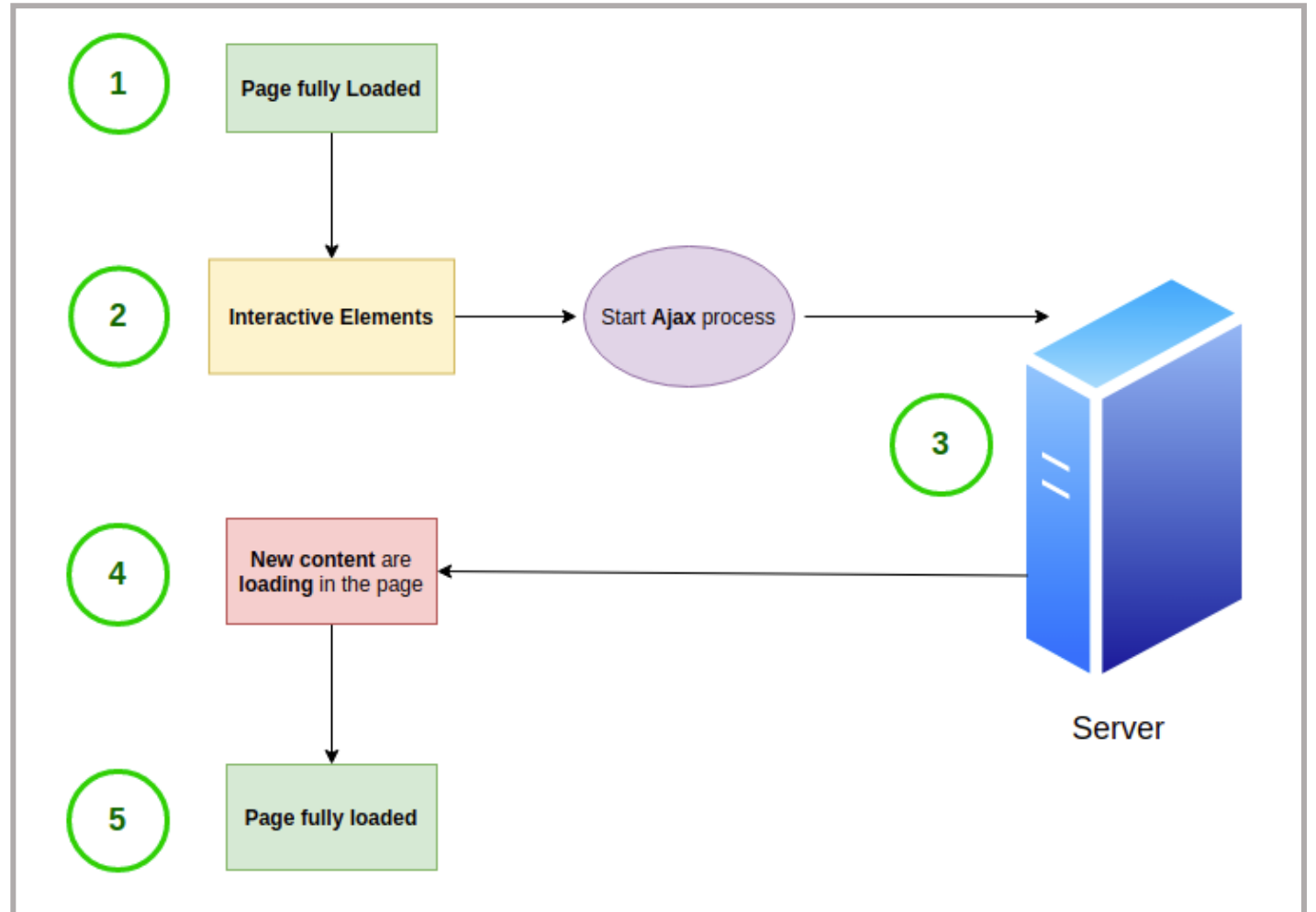
•

# Interactive elements in the page

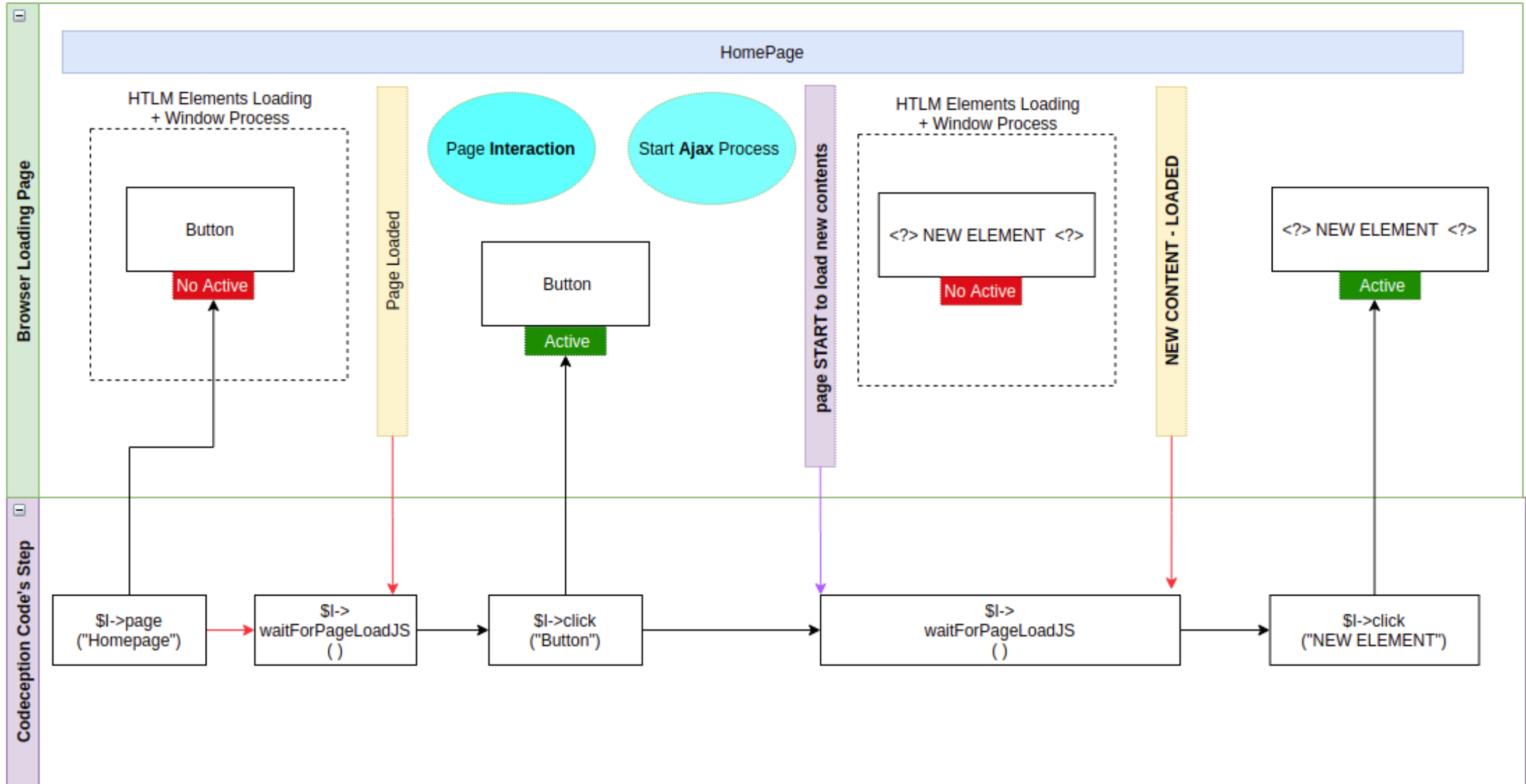
1. After loaded fully the page , When you interact with elements that will launch ajax proces , there will be loaded new content ( **starting a new loading process** ) in a part of the page .

2. in the loading process of the new content , we have to stop the Codeception code untill it is finished .

3 . When the new content si fully loaded , you can excecute correctly the next codeception code



# Interactive Elements in the



# Example Interactive page form

---

Max BTC (tcp/udp) bandwidth [Gbps]  valid until [i](#)

---

▼ Services Credentials

[ADD](#)

---

1. In this form , if you press " ADD " button , you'll create another set of fields and a little " trash " button .

When you press " ADD " will start the ajax process and new page's content will be loaded .

# Example Interactive page form

Max BTC (tcp/udp) bandwidth [Gbps]

valid until 

## ▼ Services Credentials

Username

Password

Type to change



ADD

2. you need to wait that the new content will be fully loaded before press the " trash " button . Otherwise it will be not active .

# AcceptanceHelper Functions

- Function : **waitAjaxLoad** ( \$timeout )

This function will catch all the Ajax's process and wait until they are terminate

.

- Function : **waitPageLoad** ( \$timeout )

This function will :

1. Wait until the page's content are loaded ( documentReady == complete )
2. Will launch the " waitAjaxLoad " function to load the Ajax process correctly

- Function : **amOnPageJS** ( \$link , \$timeout )

This function will :

1. Write the url's page on the browser
2. Will launch the "waitPageLoad " function to fully load the page correctly

# AcceptanceHelper Functions

- Function **checkNextPageUrlJS** ( \$XPath , \$link , \$timeout )

This function will :

1. Wait untill the \$XPath element will be loaded ( as HTML element )
2. Check the current browser link ( as \$link )
3. Will launch the function "waitPageLoad"

- Function **checkNextPageJS** ( \$XPath , \$timeout )

This function will :

1. Wait untill the \$XPath element will be loaded ( as HTML element )
2. Will launch the function "waitPageLoad"

# AcceptanceHelper Functions

- All the AcceptanceHelper's functions belongs by one \$timeout attribute :

- 

const **WAITFORJS\_TIMEOUT**

*Esempio : function waitPageLoad( \$timeout = self::WAITFORJS\_TIMEOUT )*

**WAITFORJS\_TIMEOUT** is an int value ( seconds ) that decide the standard timeout's value for all the AcceptanceHelper's functions .

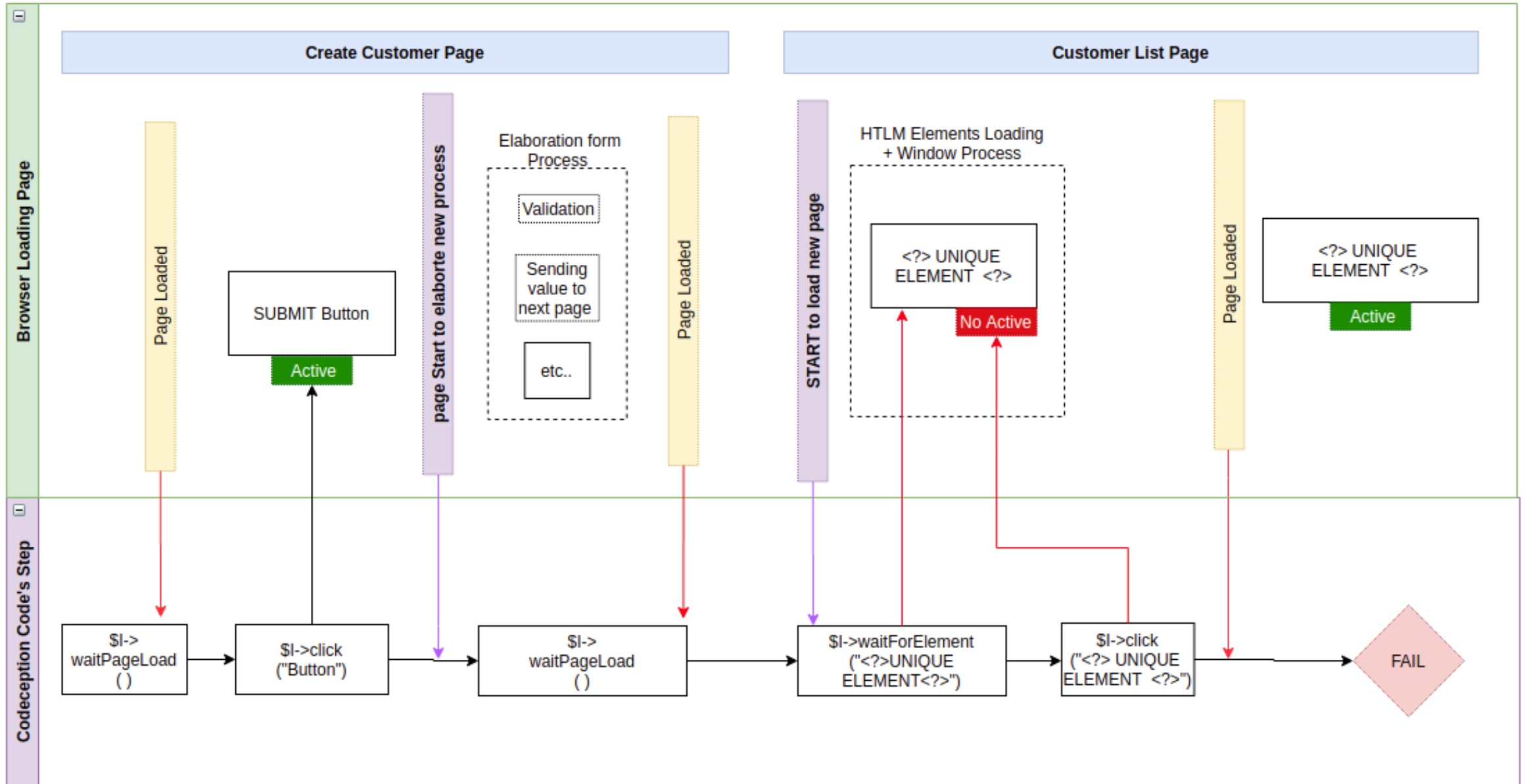
You can modify this value in base of your environment's performance :

- increment this value , if your environment has lower-performance
- reduce this value , if your enviroment has higher-performance

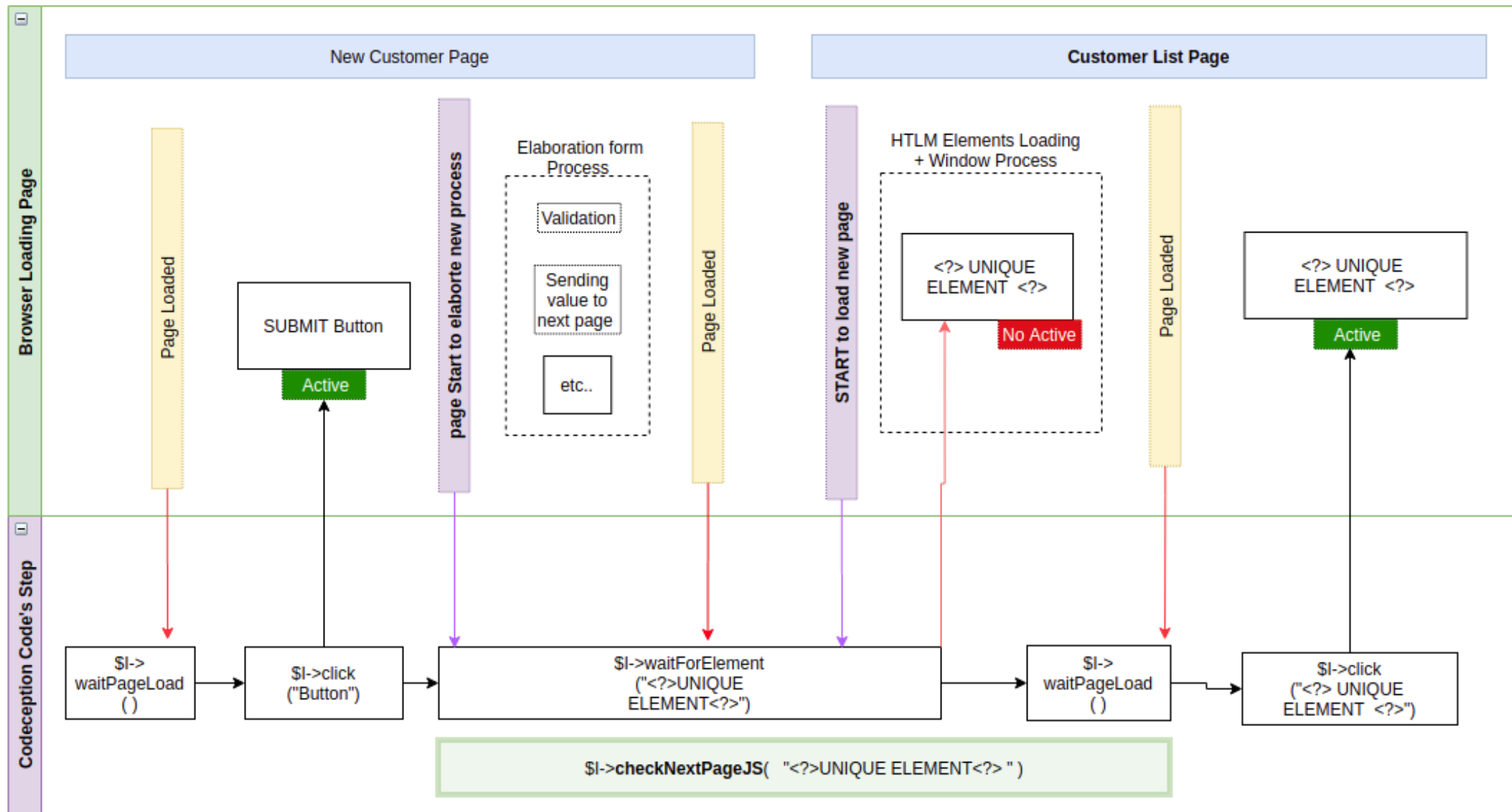
If you want to put a different timeout value for a specific situation , you have to put an int as the last param value .



# checkNextPageJS - 1 - error loading



# checkNextPageJS - 2 - loading correctly



# checkNextPageJS - 1 - error loading

We need to launch in the right time the "waitPageLoad" function

**In case - 1 - error loading :**

In page " **Create Customer Page** "

1. *after use " \$I->click("button") we directly launch "\$I->waitPageLoad()".*

This doesn't work because we are still in the "Homepage" and after clicking the "SUBMIT BUTTON" we need to load all the submit's process page ( like validation etc.. ) .

In page " **Customer List Page** "

2. *We try to interact with the next page ( but we just entered without load it completely ) and we fail because we are going to do action in the page with elements that are still " not active " .*

For this reason the test will fail

# checkNextPageJS - 2 - load correctly

In case - 2 - error loading :

- In page " **Create Customer Page** "

1. after use " `$I->click("button")` we launch a relative wait "`$I->waitForElement('<?>UNIQUE ELEMENT<?>')`".

**Unique Element** : is an element that is present only in the page where the "Submit button" will lead after we click it .

This relative wait will manage all the loading process :

- all the process in the " Create Customer Page " that start after we click the " submit button "
- All the HTML loading process , in the next page " Customer List Page " that we need to load the "UNIQUE ELEMENT " .

- In page " **Customer List Page** "

**Important** : With the " relative wait " that point to an unique element in this page , we are sure that we are in the " Customer List Page "

2. Now the command "`$I->waitPageLoad()`" will start and will wait until the " Customer List Page " will be loaded fully correctly .

3. From this point we can interact with the page .

Is very important to launch , in the correct time , these relative waits . Because if you launch before or after

**Any  
Questions ?**

---

