


```markdown

**\*\* Test Environment:\*\***

- Browser: [Google Chrome Versión 120.0.6099.199 (Build oficial) (x86\_64)]
-  Operating System: [ OS Catalina 10.15.7]

**\*\*  Preconditions:\*\***

The web page "https://the-internet.herokuapp.com/" is accessible.

**\*\*Color Legend\*\***

Request for Review

Pass

Fail

**\*\*  QA Tester:\*\***

Santiago\_J\_Montes\_de\_Oca

**\*\*Test Case ID:\*\* TC\_2AddRemoveElements\_001**

**\*\*Test Case Title:\*\* Verify Adding and Removing Elements **

**\*\*  Test Objective:\*\***


To ensure that the "Add/Remove Elements" functionality allows users to successfully add and remove elements on the page.

**\*\*  Test Data:\*\* None**

**\*\*Test Steps:\*\***



1. Open the web page and click on [Add/Remove Elements](#) button/section.

**\*\*  Expected Result:\*\***

 The page is successfully loaded.

2. Click on the "Add Element" button.

**\*\*  Expected Result:\*\***

  An element is added on the page.

3. Click the "Delete" button.


**\*\*  Expected Result:\*\***

  The added element is successfully removed.

**\*\* Test Data Variations:\*\* None**

**\*\*Postconditions:\*\***

The page should remain in a stable state, with the added element removed upon clicking "Delete."

\*\* Date:\*\* 15/01/23

\*\*Test Outcome:\*\*

Pass




\*\* Notes and Comments:\*\*[None]

\*\* Attachments:\*\*[None]

\*\* Test Case Author:\*\*

Santiago\_J\_Montes\_de\_Oca

\*\* Date:\*\*19/01/24

```

```markdown

\*\*Test Case ID:\*\* TC\_3BasicAuth\_001

\*\*Test Case Title:\*\* Verify Basic Authentication Functionality 

\*\* Test Objective:\*\*

To ensure that the "Basic Auth" functionality allows users to authenticate successfully using basic authentication credentials.

\*\* Test Data:\*\*

- Username: [admin]

- Password: [admin]

\*\*Test Steps:\*\*

1. Accessing the Basic Auth section.

- Open the web page and click on the "Basic Auth" link.

\*\* Expected Result:\*\*

The browser's authentication dialog is prompted.

2. Entering valid credentials.

- Enter the valid username and password in the authentication dialog.

\*\* Expected Result:\*\*


The user is successfully authenticated, and the protected content is accessible.


\*\* Test Data Scenarios:\*\*

- \*Scenario 1: Incorrect Username\*

- Username: [invalid\_username]

- Password: [valid\_password]

- \*\* Expected Result:\*\*

 Authentication fails, and access is denied.

- \*Scenario 2: Incorrect Password\*

- Username: [valid\_username]
- Password: [invalid\_password]

- \*\*👁 Expected Result:\*\*

❌ Authentication fails, and access is denied.

\*\*Postconditions:\*\*

The protected content should be accessible after successful authentication.

\*\*Test Outcome:\*\*

Review the appropriate execution approach.

\*\*📝 Notes and Comments:\*\*

Cypress does not allow the manipulation of pop-up windows; it would be necessary to configure the JavaScript code. I leave here an approach:

```
it('Manejo de ventana emergente con usuario y contraseña', () => {
 cy.window().then((win) => {
 // Simular entrada de texto en el prompt para el usuario y la contraseña
 cy.stub(win, 'prompt').onFirstCall().returns('admin').onSecondCall().returns('admin');
 });
```

```
 // Hacer clic en un elemento que desencadenará la ventana emergente
 cy.get('#boton-abre-popup').click();
```

```
 // Aceptar la ventana emergente presionando "Enter"
 cy.get('body').type('{enter}');
```

```
 // Puedes realizar más acciones después de la aceptación si es necesario
});
```

\*\*📎 Attachments:\*\*[None]

\*\*👤 Test Case Author:\*\*

Santiago\_J\_Montes\_de\_Oca

\*\*📅 Date:\*\*19/01/24

``

``markdown

\*\*Test Case ID:\*\* TC\_4BrokenImages\_001

\*\*Test Case Title:\*\* Verify Handling of Broken Images 🚫🖼

\*\*🎯 Test Objective:\*\*

To confirm that the "Broken Images" section displays appropriate behavior when encountering broken images.

\*\*📊 Test Data:\*\* None

\*\*Test Steps:\*\*

1. Opening website section.

- Open the web page and click on the "Broken Images" link.

\*\*👁 Expected Result:\*\*

✅ The "Broken Images" section is successfully loaded.

2. Verify the presence of images.

- Check if there are images displayed in the section.

\*\*👁 Expected Result:\*\*

✅ Images are present and displayed on the page.

\*\* Test Data Variations:\*\* None

\*\*Postconditions:\*\*

The page should remain in a stable state, with appropriate handling of broken images.

\*\*Test Outcome:\*\*

Pass ✅

\*\*📝 Notes and Comments:\*\* [None]

\*\*📎 Attachments:\*\* [None]

\*\*👤 Test Case Author:\*\*

Santiago\_J\_Montes\_de\_Oca

\*\*📅 Date:\*\* [Insert Date]

...

---

\*\*Test Case ID:\*\* TC\_5ChallengingDOM\_001

\*\*Test Case Title:\*\* Verify Challenging DOM with Unique IDs, Table, and Canvas Elements

\*\*🎯 Test Objective:\*\*

To ensure that the "Challenging DOM" section functions correctly, considering unique IDs, a table with no helpful locators, and a canvas element.

\*\*📊 Test Data:\*\* None

\*\*Test Steps:\*\*

1. Opening challenging DOM section.

- Open the web page and navigate to the "Challenging DOM" section.

\*\*👁 Expected Result:\*\*

✅ The "Challenging DOM" section is successfully loaded.

2. Verify unique IDs.

- Check for unique classes or IDs among the challenging elements (e.g., buttons, etc.).

\*\*👁 Expected Result:\*\*

- ✓ - There are:
  - 3 buttons in the same column. Class: `.button`
  - 11 rows and 11 cells in the table. Class: `.large-10.columns`
  - Canvas Element: Id: `#canvas`

### 3. Verify the canvas element.

- Examine the canvas element within the "Challenging DOM" section.

\*\*👁 Expected Result:\*\*

- ✓ The canvas element is present and can be identified.

### 4. Interact with challenging elements.

- Interact with each challenging element, considering the unique IDs, table, and canvas.

\*\*👁 Expected Result:\*\*

- ✓ Challenging elements respond as expected.

\*\* Test Data Variations:\*\* None

\*\*Postconditions:\*\*

The "Challenging DOM" page should remain in a stable state, with elements responding as expected.

\*\*Test Outcome:\*\*

Pass ✓

\*\*📝 Notes and Comments:\*\* None

\*\*📎 Attachments:\*\* [No attachments]

\*\*👤 Test Case Author:\*\*

Santiago\_J\_Montes\_de\_Oca

\*\*📅 Date:\*\* 21/01/24

```markdown

Test Case ID: TC_6Checkboxes_001

Test Case Title: Verify Checking and Unchecking Checkboxes ✓📝

🎯 Test Objective:

To ensure that the "Checkboxes" functionality allows users to successfully check and uncheck checkboxes on the page.


📊 Test Data: None

Test Steps:

1. Opening website section.

- Open the web page and click on the "Checkboxes" button/section.



****👁 Expected Result:****

 The page is successfully loaded.

2. Check the first checkbox.

- Click on the first checkbox.


****👁 Expected Result:****

  The first checkbox is successfully checked.

3. Uncheck the second checkbox.

- Click on the second checkbox.

****👁 Expected Result:****



  The second checkbox is successfully unchecked.

****Test Data Variations:**** None



****Postconditions:****

The page should remain in a stable state, with checkboxes reflecting their checked or unchecked status.

****Test Outcome:****

[Pass / Fail / Not Executed]   

****📝 Notes and Comments:****

-  Ensure that checkbox actions are responsive.
-  Check for any error messages or unexpected behaviors.

****📎 Attachments:****

[No attachments]

****👤 Test Case Author:****

Santiago_J_Montes_de_Oca

****📅 Date:****

July 17