```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.

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 IDs among the challenging elements (e.g., buttons, checkboxes).
   👁️ Expected Result:
```

✅ There are three (3) buttons in the same column (div).

3. Verify the presence of a table with no helpful locators.
   - Inspect the table within the "Challenging DOM" section and assess the absence of helpful locators.
   👁️ Expected Result:
   ✅ The table exists, and there are no identifiable locators.

4. Verify the canvas element.
   - Examine the canvas element within the "Challenging DOM" section.
   👁️ Expected Result:
   ✅ The canvas element is present and can be identified.

5. 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 / Fail / Not Executed] ✅ ❌

📝 Notes and Comments:
- Verify that unique IDs are indeed well-named and unlikely to change.
- Check for any unexpected behaviors related to the table and canvas elements.

📎 Attachments:
[No attachments]

👤 Test Case Author:
[Your Name]
📅 Date: [Fecha]

```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: [insertar fecha] 📅
~~~

```markdown
**Test Case ID:** TC_7ContextMenu_001

**Test Case Title:** Verify Context Menu Functionality 🍲
~~~

**🎯 Test Objective:**
To ensure that the "Context Menu" functionality allows users to interact with the context menu on the page.

**📊 Test Data:** None

**Test Steps:**

1. Opening the Context Menu section.
   - Open the web page and click on the Context Menu button/section.
   **👁️ Expected Result:**
   ✅ The page is successfully loaded.

2. Right-click on the context menu area.
   - Right-click on the designated context menu area.

   **👁️ Expected Result:**
   ✅ A context menu is triggered and displayed.

3. ** Step Description:** Verify options in the context menu.
   - Check for the presence of standard options in the context menu (e.g., Copy, Paste, etc.).

   **👁️ Expected Result:**
   ✅ The context menu contains expected options.

** Test Data Variations:** None

**Postconditions:**
The page should remain in a stable state, with the context menu options functioning as expected.

**Test Outcome:**

[Pass / Fail / Not Executed]✅ ❌ 📺

**📝 Notes and Comments:**
- 🚀 Ensure that the context menu response is prompt.
- 🧐 Check for any error messages or unexpected behaviors.

**📎 Attachments:**
[No attachments]

**👤 Test Case Author:**
Santiago_J_Montes_de_Oca
**📅 Date:**[ Insertar fecha] 📅
~~~

---

```markdown
~~~

**Test Case ID:** TC_8DigestAuthentication_001
**Test Case Title:** Verify Digest Authentication 🌐

**🎯 Test Objective:**
To ensure that the "Digest Authentication" functionality works correctly, allowing users to authenticate successfully.

**📊 Test Data:**
- Username: [Provided username for authentication]
- Password: [Provided password for authentication]

**Test Steps:**

1. Opening Digest Authentication section.
   - Open the web page and click on the "Digest Authentication" link.
   **👁️ Expected Result:**
   ✅ The login prompt for Digest Authentication is displayed.

2. Enter valid credentials.
   - Enter the provided username and password in the authentication prompt.
   **👁️ Expected Result:**
   ✅ Credentials are accepted, and access is granted.

3.Verify successful login.
   - Ensure that after entering valid credentials, the user is successfully authenticated.
   **👁️ Expected Result:**
   ✅ User gains access to the Digest Authentication protected content.

** Test Data Variations:**
- Test the scenario with an invalid username or password to ensure proper authentication failure.

**Postconditions:**
The page should remain in a stable state, with the user authenticated and granted access to the protected content.

**Test Outcome:**

[Pass / Fail / Not Executed] ✅ ❌ 📺

**📝 Notes and Comments:**
- 🚀 Ensure that the authentication process is prompt.
- 🧐 Check for any error messages or unexpected behaviors.

**📎 Attachments:**
[No attachments]

**👤 Test Case Author:**
Santiago_J_Montes_de_Oca
**📅 Date:** [Insert Date]

---

```markdown
**Test Case ID:** TC_9DisappearingElements_001
**Test Case Title:** Verify Adding and Removing Elements 🧩

**🎯 Test Objective:**
To ensure that the "Disappearing Elements" functionality allows users to successfully add and remove elements on the page.

**📊 Test Data:** None

**Test Steps:**

1. Opening website section.
   - Open the web page and click on the "Disappearing Elements" button/section.
   **👁️ Expected Result:**
   ✅ The page is successfully loaded.

2.Verify the presence of initial elements.
   - Check for the presence of existing elements on the page.
   **👁️ Expected Result:**
   ✅ The initial elements are visible.

3. Click the "Add Element" button.
   - Click on the "Add Element" button.
   **👁️ Expected Result:**
   ✅➕ An additional element is added on the page.

4. ** Step Description:** Verify the presence of the added element.
   - Check for the presence of the newly added element.
   **👁️ Expected Result:**
   ✅ The added element is visible.

5. ** Step Description:** Click the "Refresh" button.
   - Click on the "Refresh" button to reload the page.

   **👁️ Expected Result:**
   ✅ The page is refreshed without errors.

6. ** Step Description:** Verify the disappearance of the added element after refresh.
   - Check if the previously added element is no longer visible.
   **👁️ Expected Result:**
   ✅➖ The added element is not visible after refreshing.
```

** Test Data Variations:** None

Postconditions:
The page should remain in a stable state, with the added element no longer visible after the refresh.

Test Outcome:
[Pass / Fail / Not Executed] ✅ ❌ 📺

📝 Notes and Comments:
- 🚀 Ensure that the page response is prompt.
- 🧐 Check for any error messages or unexpected behaviors.

📎 Attachments:
[No attachments]

👤 Test Case Author:
Santiago_J_Montes_de_Oca
📅 Date: [Current Date]
```

---
```