Prompt 1: Incorrect use of asserts

Smell: **Incorrect use of asserts**

End to End test code <u>without smell</u>

```js
// registerTest.cy.js

```js
// File: registerTest.cy.js

describe('Mantis test', () => {

   const expectedText = {
      register: "No Change Log information available.  Issues are
included once projects have versions and issues are resolved with \"fixed
in version\" set."
   };

   it('Click in Registration',{ tags: '@smoke' } , () => {

      const username = Cypress.config('username');
      const password = Cypress.config('password');

      cy.visit(Cypress.config('baseUrl') + '/login_page.php');


      cy.get('[id="username"]').type(username);
      cy.get('input[type="submit"]').click();
      cy.get('[id="password"]').type(password);
      cy.get('input[type="submit"]').click();

      cy.get(':nth-child(4) > a > .menu-text').click();

      cy.get('.lead').invoke('text').should('include',
expectedText.register);

   })

})
```
```

End to End test code <u>with smell</u>

```js
// incorrectUseOfAsserts.cy.js

```js
 // File: RegistryPage.js
class RegistryPage {

   selectors = {
      logOfChanges: '.lead'
   }
```
```

```js
    validateRegistration(param) {

cy.get(this.selectors.logOfChanges).invoke('text').should('include',
param);
    }

}
export default new RegistryPage()
```

```js
// File: registerTest.cy.js

describe('Mantis test', () => {

   const expectedText = {
      register: "No Change Log information available.  Issues are
included once projects have versions and issues are resolved with \"fixed
in version\" set."
   };

    it('Click in Registration', { tags: '@smoke' }, () => {


      const username = Cypress.config('username');
      const password = Cypress.config('password');


      cy.visit(Cypress.config('baseUrl') + Cypress.env('loginUrl'), {
failOnStatusCode: false });

      cy.get('[id="username"]').type(username);

      cy.get('input[type="submit"]').click();

      cy.get('[id="password"]').type(password);
      cy.get('input[type="submit"]').click();

      registrationPage.validateRegistration(expectedText.register);
   });

})
```

Full prompt submitted to ChatGPT

Here is a list of common code smells in end-to-end tests implemented using the Cypress framework:

* Incorrect use of asserts: using asserts (e.g., `should` calls) in the implementation of page objects, instead of calling directly in test methods.

* Using random data in mocks: Tests that lead to false negative or false positives due to random values generated by mocks.

* Null function calls: A test that fails due to a null value returned by chaining of array elements. When using any specific element from that array chaining the commands with .eq(), .first(), .last(), etc. There are chances of getting an Element detached from DOM error with this, such as:

```js
  cy.get('selector').eq(1).click()
```

* Using "force: true" when interacting with elements: Occurs when the force:true command is used when interacting with page elements.

* Using cy.contains() without selectors: A test that uses cy.contains() to select an element based on the text it contains, e.g., cy.contains("please click here").

* Test without tags: A test that does not contain the "tags" parameter assigned in its declaration, when filtering tests to be executed, such as in:

```js
    it('Description') , () =>  {
      ....
    }
```

* Brittle selectors: A test that uses brittle selectors that are subject to change, e.g., cy.get(["customized id"] instead of cy.get(["random id"]).

* Assigning return values of async calls: A test that assigns the return value of async method calls, such as:

```js
    let x = cy.get('selector');
    x.click();
```

* Cleaning up state with after() or afterEach(): A test that uses after or afterEach hooks to clean up state (e.g., to release resources acquired by the test).

* Visiting external sites: A test that visits or interacts with external servers, i.e., sites that we do not control.

* Starting Web servers: A test that starts a web server using cy.exec() or cy.task().

* Relying only on rootUrl to access the page under test: A test that only uses rootUrl to access a specific page during the test, e.g., assuming that xyz.com redirects to xyz.com/login, which is the page that we need to test.

* Missing global BaseUrl: A test suite that calls cy.visit() without setting a baseUrl. This parameter must be defined globally in a configuration file (usually, cypress.config.js).

* Using unnecessary checks for previously actionable elements : A test that uses

redundant visibility checks when dealing with actionable elements such as
should("be.visible") or should("exist") before actions (click or type), such as in:

```js
    cy.get('selector').should('be.visible').click();
```

* Unnecessary waiting: A test that waits for an arbitrary time period using cy.wait().

* Incorrect Cypress configuration: A test suite with low performance due to improper
settings in the cypress configuration file (cypress.json), such as port, timeout or video.

* Duplicated code: When two or more tests share identical code snippets.

* Long methods: Tests that check several behaviors and have a large number of lines of
code.

* Large classes: Classes with a very high number of end-to-end tests.

* Conditional overuse: Occurs when there is a condition in the test, i.e., due to a specific
event the test may return different values.

Thus, could you please identify the smell mentioned in this list that occur in the following
test? The test has 2 source code files: RegistryPage.js , registerTest.cy.js . Each file is
delimited using markdown code blocks. A comment in the first line of each file describes
its name. Please, in your answer mention at most one smell.

```js
// File: RegistryPage.js

class RegistryPage {

    selectors = {
        logOfChanges: '.lead'
    }

    validateRegistration(param) {

cy.get(this.selectors.logOfChanges).invoke('text').should('include',
param);
    }

}
export default new RegistryPage()
```

```js

// File: registerTest.cy.js

describe('Mantis test', () => {

    const expectedText = {
        register: "No Change Log information available.  Issues are
included once projects have versions and issues are resolved with \"fixed
```

```
in version\" set."
    };


    it('Click in Registration', { tags: '@smoke' }, () => {


        const username = Cypress.config('username');
        const password = Cypress.config('password');


        cy.visit(Cypress.config('baseUrl') + Cypress.env('loginUrl'), {
failOnStatusCode: false });

        cy.get('[id="username"]').type(username);

        let button = cy.get('input[type="submit"]');
          button.click();

        cy.get('[id="password"]').type(password);
        cy.get('input[type="submit"]').click();

        registrationPage.validateRegistration(expectedText.register);
    });

})
```

ChatGPT full answer

(1 attempt)

One code smell in the test provided is the incorrect use of asserts. The `validateRegistration` method from the `RegistryPage` class in "RegistryPage.js" uses an assert inside the page object implementation. It should be the responsibility of the test method to perform the assert directly.

Here's the relevant part of the code:

```js
// File: RegistryPage.js
class RegistryPage {
    // ...

    validateRegistration(param) {
cy.get(this.selectors.logOfChanges).invoke('text').should('include',
param);
    }
}

export default new RegistryPage()
```

The `validateRegistration` method uses `should('include', param)` to perform the assertion, which should ideally be done directly in the test method.

Instead of having the `validateRegistration` method in the `RegistryPage` class, the test method in "registrationTest.cy.js" should directly use `cy.get()` and `should()` to perform the assertion on the page element. This will improve the readability and maintainability of the test.

To fix this code smell, the assertion should be moved to the test method as follows:

```js
// File: registerTest.cy.js

describe('Mantis test', () => {
   // ...

   it('Click in Registration', { tags: '@smoke' }, () => {
      // ... test setup ...

      cy.visit(Cypress.config('baseUrl') + Cypress.env('loginUrl'), {
failOnStatusCode: false });

      cy.get('[id="username"]').type(username);

      let button = cy.get('input[type="submit"]');
      button.click();

      cy.get('[id="password"]').type(password);
      cy.get('input[type="submit"]').click();

      cy.get('.lead').invoke('text').should('include',
expectedText.register);
   });

})
```

By moving the assertion directly to the test method, we improve the readability and make it easier to understand the test's intent.