1. Issues in login.test.ts

1.1. Using hardcoded data

```
it("login with valid credentials", function () {
  loginPage.login("testautomation@cypresstest.com", "Test@1234");
  myAccountPage.validateSuccessfulLogin();
  myAccountPage.logout();
  myAccountPage.validateSuccessfulLogout();
});
```

1.2. The code is not formatted.

The code is not formatted. There are no extra lines or unnecessary entries, but the code contains long lines and is therefore difficult to read.

Before formatting the code:

```
ypress > e2e > tests > TS login.test.ts > ...
4   describe('Login Functionality', () => {
5       beforeEach(() => {
10      })
11      it('login with valid credentials', function () {
12          loginPage.login("testautomation@cypresstest.com", "Test@1234")
13          myAccountPage.validateSuccessfulLogin()
14          myAccountPage.logout()
15          myAccountPage.validateSuccessfulLogout()
16      })
17      it('login with valid credentials read data from fixture', function () {
18          loginPage.login(this.data.valid_credentials.emailId, this.data.valid_credentials.password)
19          myAccountPage.validateSuccessfulLogin()
20          myAccountPage.logout()
21          myAccountPage.validateSuccessfulLogout()
22      })
23      it('login with invalid email credentials read data from fixture', function () {
24          loginPage.login(this.data.invalid_credentials.invalid_email.emailId,
25              this.data.invalid_credentials.invalid_email.password)
26          loginPage.validateLoginError('Authentication failed.')
27      })
```

After formatting the code :

```
4   describe("Login Functionality", () => {
5     beforeEach(() => {
7       cy.fixture("users.json").then(function (data) {
9       });
10    });
11    it("login with valid credentials", function () {
12      loginPage.login("testautomation@cypresstest.com", "Test@1234");
13      myAccountPage.validateSuccessfulLogin();
14      myAccountPage.logout();
15      myAccountPage.validateSuccessfulLogout();
16    });
17    it("login with valid credentials read data from fixture", function () {
18      loginPage.login(
19        this.data.valid_credentials.emailId,
20        this.data.valid_credentials.password
21      );
22      myAccountPage.validateSuccessfulLogin();
23      myAccountPage.logout();
24      myAccountPage.validateSuccessfulLogout();
25    });      You, 2 days ago • Uncommitted changes
26    it("login with invalid email credentials read data from fixture", function () {
27      loginPage.login(
28        this.data.invalid_credentials.invalid_email.emailId,
29        this.data.invalid_credentials.invalid_email.password
```

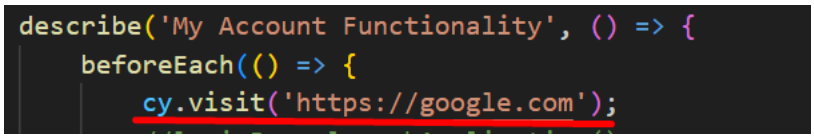
1.3. For testing field using always the same testing data.

```
it("login with valid credentials read data from fixture", function () {
  loginPage.login(
    this.data.valid_credentials.emailId,
    this.data.valid_credentials.password
  );
  myAccountPage.validateSuccessfulLogin();
  myAccountPage.logout();
  myAccountPage.validateSuccessfulLogout();
});      You, 2 days ago • Uncommitted changes
```

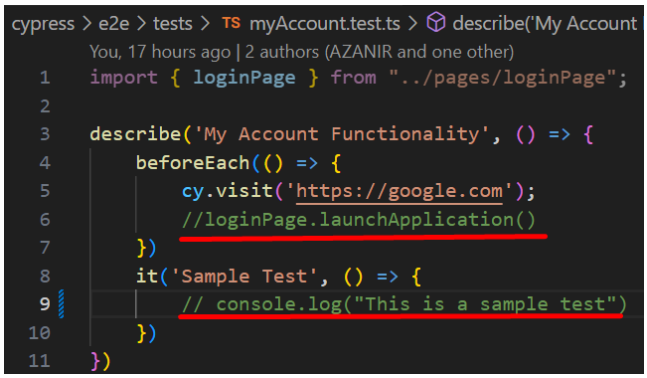
Tests load a fixed set of data located in the json file. It is good to put login and password of existing user in json file. For negative scenarios, it is better to use randomly generated values by node package faker-js for example. This also will prevent the emergence of the Pesticide Paradox.

2. Issues in myAccount.test.ts

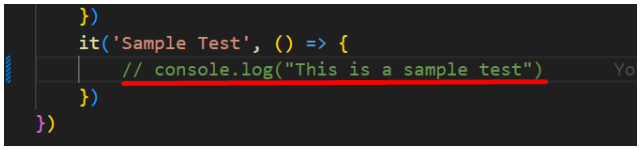
2.1. Visit a URL right in the test. Visit the URL other than the base URL.



2.2. There is commented code in the test



2.3. Using output to the console, instead of verification with expect (or assert).



3. Issues in myAccountPage.ts

3.1. Method `validateSuccessfulLogout()` should be in the loginPage.ts

Methods and getters must be located on the page to which a particular selector belongs. This **simplify maintenance** by capturing element selectors in one place and create reusable code to avoid repetition.

1. Repo has many redundant files

The folder “docs” and its content. This folder should be added to .gitignore file.