

1. Issues in login.test.ts

1.1. Using hardcoded data

```
it("login with valid credentials", function () {  
    loginPage.login("testautomation@cypressstest.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:

```
press > e2e > tests > TS login.test.ts > ...  
4 describe('Login Functionality', () => {  
5     beforeEach(() => {  
6     })  
7     it('login with valid credentials', function () {  
8         loginPage.login("testautomation@cypressstest.com", "Test@1234")  
9         myAccountPage.validateSuccessfulLogin()  
10        myAccountPage.logout()  
11        myAccountPage.validateSuccessfulLogout()  
12    })  
13    it('login with valid credentials read data from fixture', function () {  
14        loginPage.login(this.data.valid_credentials.emailId, this.data.valid_credentials.password)  
15        myAccountPage.validateSuccessfulLogin()  
16        myAccountPage.logout()  
17        myAccountPage.validateSuccessfulLogout()  
18    })  
19    it('login with invalid email credentials read data from fixture', function () {  
20        loginPage.login(this.data.invalid_credentials.invalid_email.emailId,  
21            this.data.invalid_credentials.invalid_email.password)  
22        loginPage.validateLoginError('Authentication failed.')  
23    })  
24 })  
25
```

After formatting the code :

```
4 describe("Login Functionality", () => {  
5     beforeEach(() => {  
6         cy.fixture("users.json").then(function (data) {  
7             this.data = data;  
8         });  
9     });  
10    it("login with valid credentials", function () {  
11        loginPage.login("testautomation@cypressstest.com", "Test@1234");  
12        myAccountPage.validateSuccessfulLogin();  
13        myAccountPage.logout();  
14        myAccountPage.validateSuccessfulLogout();  
15    });  
16    it("login with valid credentials read data from fixture", function () {  
17        loginPage.login(  
18            this.data.valid_credentials.emailId,  
19            this.data.valid_credentials.password  
20        );  
21        myAccountPage.validateSuccessfulLogin();  
22        myAccountPage.logout();  
23        myAccountPage.validateSuccessfulLogout();  
24    });  
25    it("login with invalid email credentials read data from fixture", function () {  
26        loginPage.login(  
27            this.data.invalid_credentials.invalid_email.emailId,  
28            this.data.invalid_credentials.invalid_email.password  
29        );  
30    });  
31 })
```

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();  
});
```

```

cypress > fixtures > {} users.json > ...
AZANIR, 3 years ago | 1 author (AZANIR)
1  {
2    .... "valid_credentials": {
3    ....   "emailId": "testautomation@cypress-test.com",
4    ....   "password": "Test@1234"
5    .... },

```

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.

```

describe('My Account Functionality', () => {
  beforeEach(() => {
    cy.visit('https://google.com');

```

2.2. There is commented code in the test

```

cypress > e2e > tests > TS myAccount.test.ts > describe('My Account
You, 17 hours ago | 2 authors (AZANIR and one other)
1  import { loginPage } from "../pages/loginPage";
2
3  describe('My Account Functionality', () => {
4    beforeEach(() => {
5      cy.visit('https://google.com');
6      //loginPage.launchApplication()
7    })
8    it('Sample Test', () => {
9      // console.log("This is a sample test")
10   })
11 })

```

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

```

    })
    it('Sample Test', () => {
      // console.log("This is a sample test")
    })
  })
}

```

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.

```

public validateSuccessfulLogout() {
  loginPage.signinBtn.should('be.visible')
}

```

1. Repo has many redundant files

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

