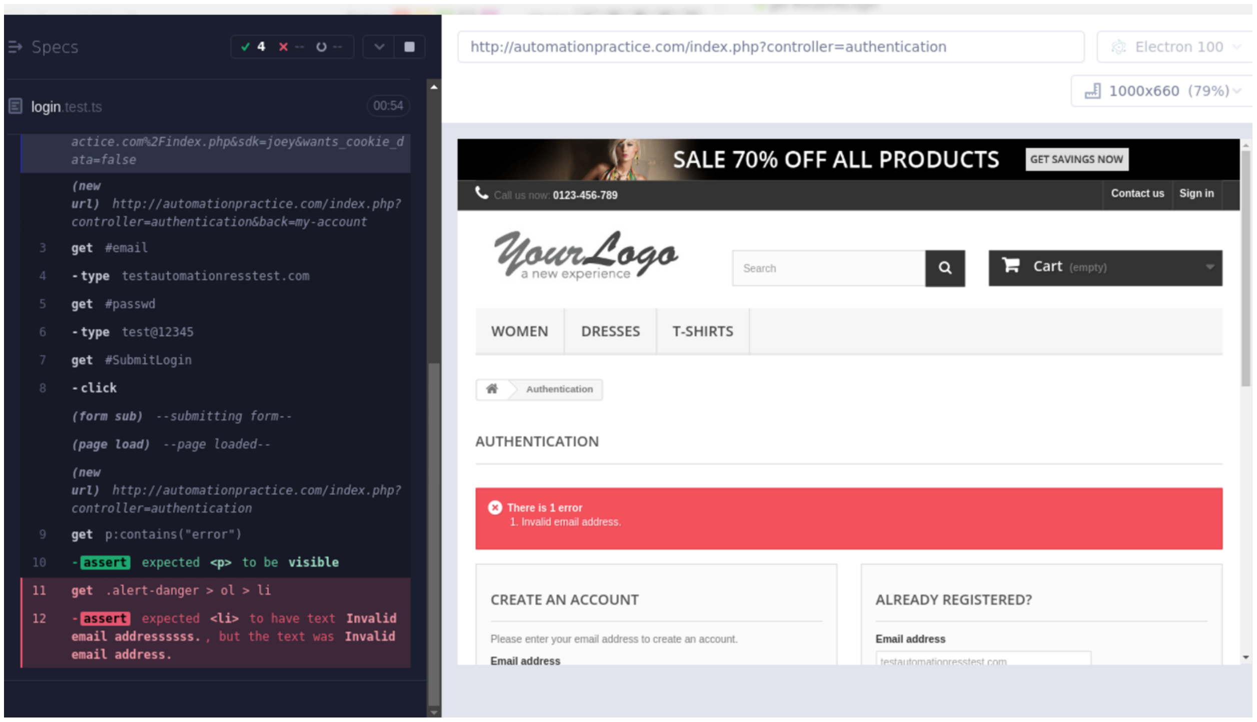
1. Test in test suite “Login Functionality” named as “login with wrong email format credentials read data from fixture” is failed

Link to this test in Allure report: https://azanir.github.io/cypressAllure/#suites/a9c8f11b4c0d702164a4731c0b55ffa0/57e1c9354671ca47/

Screenshot from allure report:



```
login.test.ts                                          00:54

actice.com%2Findex.php&sdk=joey&wants_cookie_d
ata=false
(new
url)  http://automationpractice.com/index.php?
controller=authentication&back=my-account
get  #email
-type  testautomationresstest.com
get  #passwd
-type  test@12345
get  #SubmitLogin
-click
(form sub)  --submitting form--
(page load)  --page loaded--
(new
url)  http://automationpractice.com/index.php?
controller=authentication
get  p:contains("error")
-assert  expected <p> to be visible
get  .alert-danger > ol > li
-assert  expected <li> to have text Invalid
email addressssss. , but the text was Invalid
email address.
```

http://automationpractice.com/index.php?controller=authentication

There is 1 error
1. Invalid email address.

First of all, you should check the correctness of a locator .alert-danger > ol > li.

Secondly, in the file cypress/e2e/tests/login.test.ts, we can see that everything is correct except the expected data while validating the login error. It should be “Invalid email address.” not “Invalid email addressssss.” (mistype – multiple ‘s’).

```
it('login with wrong email format credentials read data from fixture', function () {
    loginPage.login(this.data.invalid_credentials.wrong_email_format.emailId, this.data.invalid_credentials.wrong_email_format.password)
    loginPage.validateLoginError('Invalid email addressssss.')
})
```

2. Unused statement and unused import

In a file cypress/e2e/tests/myAccount.test.ts, unused commented statement exists in the beforeEach statement: //loginPage.launchApplication(), it should be removed.

```
    beforeEach(() => {
        cy.visit('https://google.com');
        //loginPage.launchApplication()
    })
```

Import of the login page is not used; it should also be removed.

```
import { loginPage } from "../pages/loginPage";
```

3. The purpose of test suite “My Account Functionality” is unclear

In a file cypress/e2e/tests/myAccount.test.ts, the test suite should be connected with account functionality according to the suit’s name, but instead, it has only one sample test that print to console message.

```
describe('My Account Functionality', () => {
    beforeEach(() => {
        cy.visit('https://google.com');
        //loginPage.launchApplication()
    })
    it('Sample Test', () => {
        console.log("This is a sample test")
    })
})
```

There are two options for improving this suite:

- If you want to test an account: Remove the Sample test and develop more tests that test My Account Functionality, such as verifying the correctness of data on the My Account page, verifying that the user can change his data or delete the account, etc.
- If you want to perform a sample test: Rename the test suite to "Sample test" and remove the beforeEach() function. Statement cy.visit('https://google.com'); could be moved to the "Sample test"; after that could come assertions about the URL of the page or title, header, or any

other element. That way, the test will test Cypress visiting the website successfully.

4. Incomplete naming or value for test data variable wrong_email_format

```
"invalid_credentials": {
    "invalid_email": {
        "emailId": "invalidUser@cypresstest.com",
        "password": "Test@1234"
    },
    "invalid_password": {
        "emailId": "testautomation@cypresstest.com",
        "password": "test@12345"
    },
    "wrong_email_format": {
        "emailId": "testautomationresstest.com",
        "password": "test@12345"
    }
```

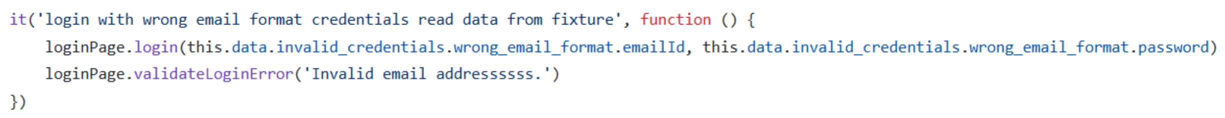
In a file cypress/fixtures/users.json in the invalid_email, invalid_password variables exist data with invalid either email or password according to the name. In the variable wrong_email_format, data with the wrong email *and* invalid password exist.

The options to fix could be:

- If the test is supposed to test the wrong email format only and a valid password: Provide a valid password in this dataset. Example of new data:

```
"wrong_email_format": {
            "emailId": "testautomationresstest.com",
            "password": "Test@1234"
        }
```

- If the test is supposed to test both wrong email format and invalid password: Rename this data set to "wrong_email_format_and_invalid_password" and rename the test "login with wrong email format credentials read data from fixture" in a file cypress/e2e/tests/login.test.ts to "login with wrong email format <u>and invalid password</u> credentials read data from fixture". That way the purpose of the test where this data is used will be to test how the application will work while the two fields have invalid values.

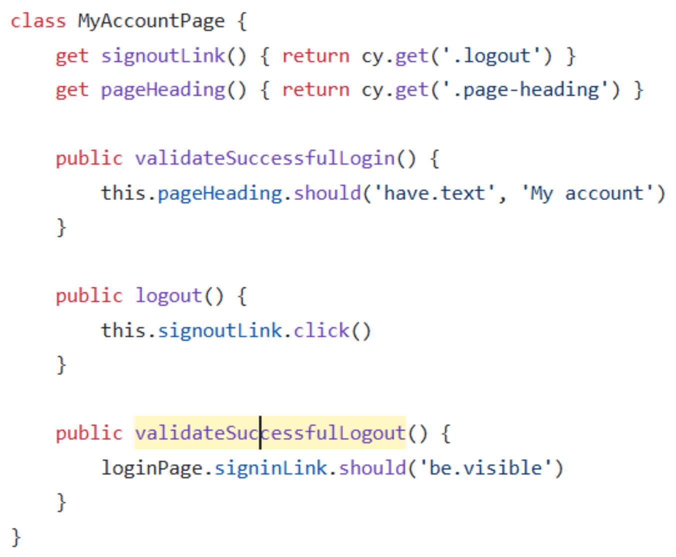


```
it('login with wrong email format credentials read data from fixture', function () {
    loginPage.login(this.data.invalid_credentials.wrong_email_format.emailId, this.data.invalid_credentials.wrong_email_format.password)
    loginPage.validateLoginError('Invalid email addressssss.')
})
```

If you don’t want to refactor this but refactor the mistype of test “login with wrong email format credentials read data from fixture” (see point 1), so probably this test will work because the application will be validating email first (as first field) -> validation will fail -> application print the error message and won’t valid the password (second field). But for clarity, I think you should provide either a valid password in this case or rename the variable to confirm that two values are invalid.

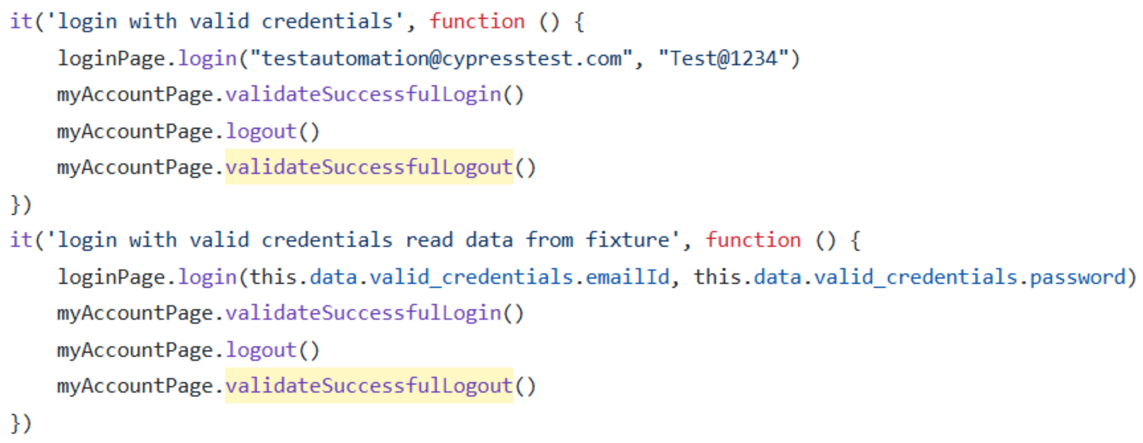
5. The function validateSuccessfulLogout is in the class MyAccountPage, but it logically should be in LoginPage

In a file cypress/e2e/pages/myAccountPage.ts function validateSuccessfulLogout interact with the element sighinLink that getting in the loginPage. This method should be moved to the class LoginPage, because it isn’t using any data of MyAccountPage.



```
class MyAccountPage {
    get signoutLink() { return cy.get('.logout') }
    get pageHeading() { return cy.get('.page-heading') }

    public validateSuccessfulLogin() {
        this.pageHeading.should('have.text', 'My account')
    }

    public logout() {
        this.signoutLink.click()
    }

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

Then you should change the reference myAccountPage to the loginPage in the file cypress/e2e/tests/login.test.ts.



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

6. Locators in MyAccountPage could be improved

In a file cypress/e2e/pages/myAccountPage.ts, two get-element methods are used to get elements by a CSS selector by class. It's a good practice to write a more specific locator that is based on a stable DOM tree structure of the page rather than a class name. It could be that the class name will be renamed, and your tests are going to fail.

```
get signoutLink() { return cy.get('.logout') }
get pageHeading() { return cy.get('.page-heading') }
```

7. Locators in LoginPage could be improved

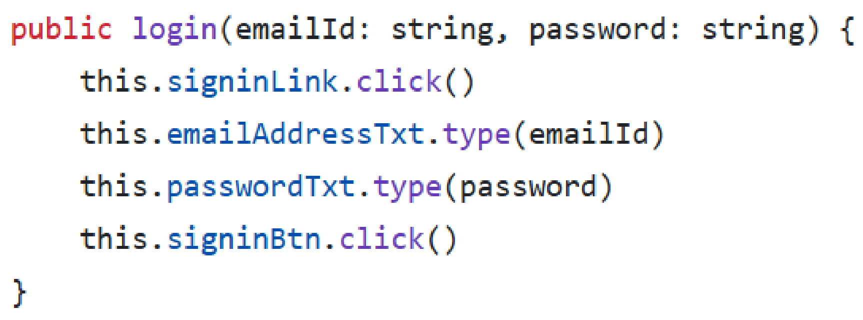
In a file cypress/e2e/pages/loginPage.ts, locators also could be improved.

```
get signinLink() { return cy.get('.login') }
get emailAddressTxt() { return cy.get('#email') }
get passwordTxt() { return cy.get('#passwd') }
get signinBtn() { return cy.get('#SubmitLogin') }
get alertBox() { return cy.get('p:contains("error")')}
get alertMessage() { return cy.get('.alert-danger > ol > li') }
```

Locator .alert-danger > ol > li is more specific than others, so it is better. Try to use something like body > form > #login or use test-ids if developers provided them for testers. Such test attributes in the tag don't change and could be used for testing purposes.

8. There is no waiting for elements to load after clicking on a sign in link

The login functionality in a file cypress/e2e/pages/loginPage.ts is performed by clicking on the link, filling and submitting the form. It is better to divide the redirecting by link and submit the form into two different functions, and wait for loading the page.



```
public login(emailId: string, password: string) {
    this.signinLink.click()
    this.emailAddressTxt.type(emailId)
    this.passwordTxt.type(password)
    this.signinBtn.click()
}
```

I can recommend you move the clicking on a sign-in link to another method and make some assertions that some specific element on the page is visible; that way, we can be sure that the page was loaded and typing will work. Example:

```
public clickingOnSigninLink(){

   this.signinLink.click()

   this.emailAddressTxt. should('be.visible')

}
```

Recommendation/question: signInLink is probably located on the main page or any other page, but not login. Maybe it would be better to move getting this element and clicking on signInLink in another class, like the Main class?

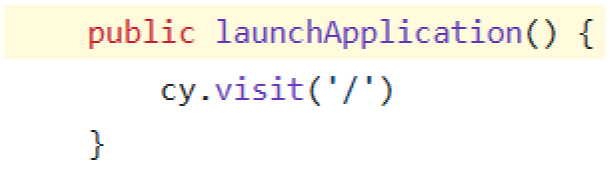
But if you want to leave interacting with these elements inside the Login class, I suppose it’s okay because it is connected with the user login journey, and the page object is not necessarily a whole page; it is a part of the interface, in which elements are logically connected together.

Filling out login the form and submitting logic could be in the method login. Example:

```
public login(emailId: string, password: string){

   this.emailAddressTxt.type(emailId)

   this.passwordTxt.type(password)

   this.signinBtn.click()

}
```

9. Launch application can be incapsulated in different class

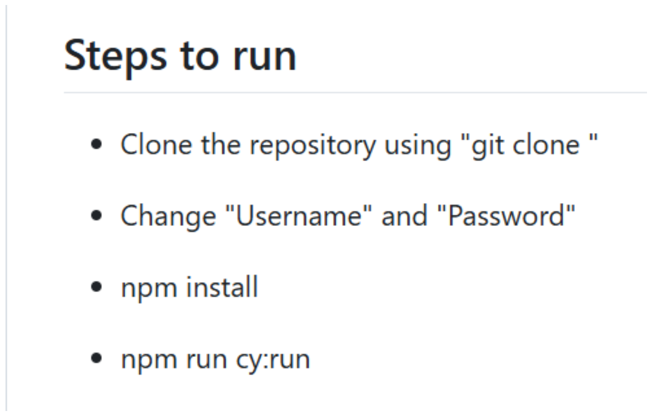
In a file, the cypress/e2e/pages/loginPage.ts function launchApplication() isn’t really connected to the login page functionality or login page elements, so this method could be moved to a BasePage class in the method named navigate().



```
    public launchApplication() {
        cy.visit('/')
    }
```

10. Provide more details in a Read.me file

It could be helpful if, in a file README.md for the step "Change Username and Password," there was a link to the file where to change username and password or more details about what these credentials are used for.



Steps to run

- Clone the repository using "git clone "
- Change "Username" and "Password"
- npm install
- npm run cy:run