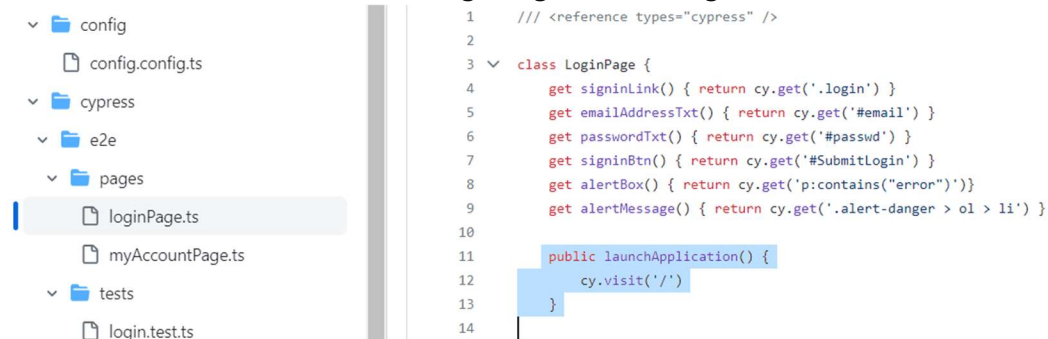


- POM: It is recommended to use two-levels Page Object Model, you might have a structure with mainPage and other pages that will extend the mainPage. loginPage and myAccountPage can inherit the methods of mainPage.

This method can be moved from loginPage to mainPage.

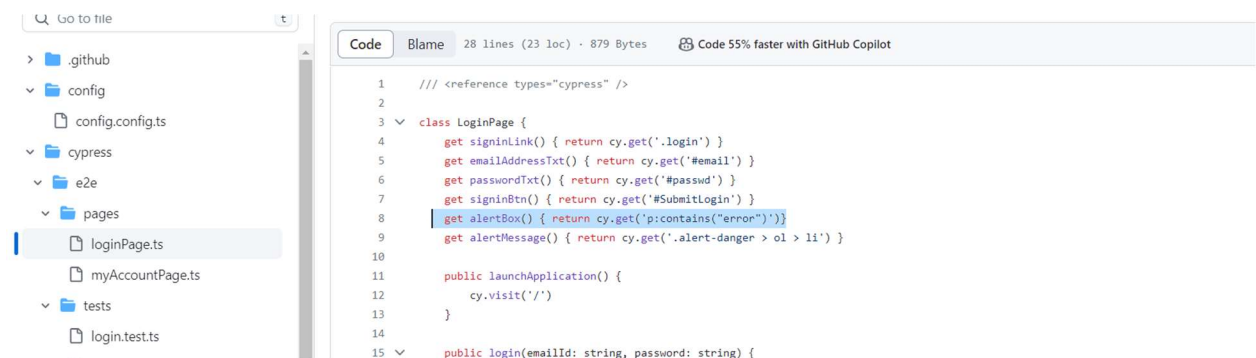


```

1  /// <reference types="cypress" />
2
3  class LoginPage {
4      get signinLink() { return cy.get('.login') }
5      get emailAddressTxt() { return cy.get('#email') }
6      get passwordTxt() { return cy.get('#passwd') }
7      get signinBtn() { return cy.get('#SubmitLogin') }
8      get alertBox() { return cy.get('p:contains("error")') }
9      get alertMessage() { return cy.get('.alert-danger > ol > li') }
10
11     public launchApplication() {
12         cy.visit('/')
13     }
14

```

- [LoginPage] It is better to avoid selectors by text. If the text changes, test will fail. In the future app might have different localizations and the selector by text will be not convenient.

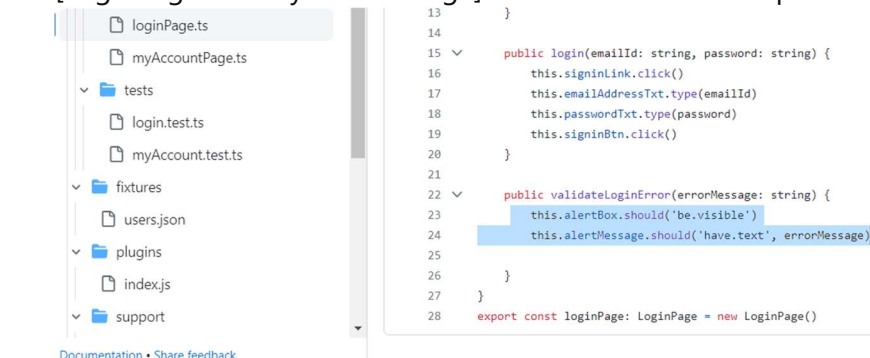


```

1  /// <reference types="cypress" />
2
3  class LoginPage {
4      get signinLink() { return cy.get('.login') }
5      get emailAddressTxt() { return cy.get('#email') }
6      get passwordTxt() { return cy.get('#passwd') }
7      get signinBtn() { return cy.get('#SubmitLogin') }
8      get alertBox() { return cy.get('p:contains("error")') }
9      get alertMessage() { return cy.get('.alert-danger > ol > li') }
10
11     public launchApplication() {
12         cy.visit('/')
13     }
14
15     public login(emailId: string, password: string) {

```

- [LoginPage and myAccountPage] It is recommended to place assertions in spec files.

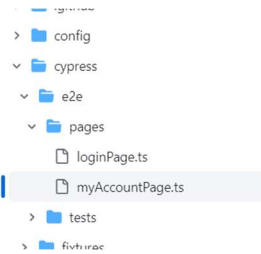


```

13     }
14
15     public login(emailId: string, password: string) {
16         this.signinLink.click()
17         this.emailAddressTxt.type(emailId)
18         this.passwordTxt.type(password)
19         this.signinBtn.click()
20     }
21
22     public validateLoginError(errorMessage: string) {
23         this.alertBox.should('be.visible')
24         this.alertMessage.should('have.text', errorMessage)
25     }
26 }
27
28 export const loginPage = new LoginPage()

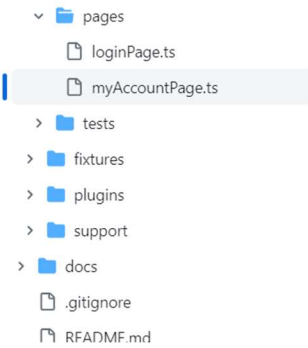
```

- It is recommended to place assertions in spec files.
- You can't import one page to another except for inheritance.



```
1  /// <reference types="cypress" />
2
3  import { LoginPage } from "../LoginPage"
4
5  class MyAccountPage {
6    get signoutLink() { return cy.get('.logout') }
7    get pageHeading() { return cy.get('.page-heading') }
8
9    public validateSuccessfulLogin() {
10     this.pageHeading.should('have.text', 'My account')
11   }
12 }
```

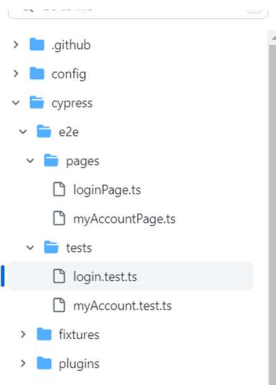
- It is recommended to place assertions in spec files.



```
6    get signoutLink() { return cy.get('.logout') }
7    get pageHeading() { return cy.get('.page-heading') }
8
9    public validateSuccessfulLogin() {
10     this.pageHeading.should('have.text', 'My account')
11   }
12
13   public logout() {
14     this.signoutLink.click()
15   }
16
17   public validateSuccessfulLogout() {
18     loginPage.signInLink.should('be.visible')
19   }
20 }
```

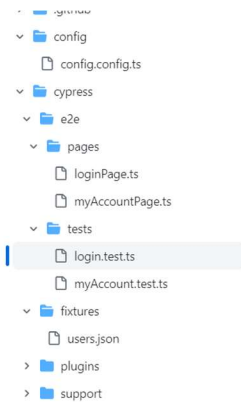
Tests file:

- It is better to not hardcode test data. You can use data from users.json file('valid_credentials').



```
2  import { MyAccountPage } from '../pages/myAccountPage'
3
4  describe('Login Functionality', () => {
5    beforeEach(() => {
6      loginPage.launchApplication()
7      cy.fixture('users.json').then(function (data) {
8        this.data = data;
9      })
10   })
11   it('login with valid credentials', function () {
12     loginPage.login("testautomation@cypress.test.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 })
```

- Test organization: it's a good practice to have one test case for one functionality. So login and logout should be 2 separate test cases. You can use 2 describe blocks in 'login.test.ts' file: one for login test cases and the other one for logout. This provides a clear structure within a spec file. If initial intention was to return system to default state, use afterHooks for this purpose.

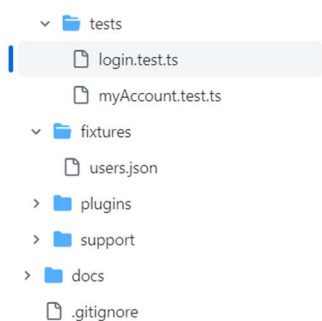


```

1 import { LoginPage } from '../pages/LoginPage'
2 import { myAccountPage } from '../pages/myAccountPage'
3
4 describe('Login Functionality', () => {
5   beforeEach(() => {
6     LoginPage.launchApplication()
7     cy.fixture('users.json').then(function (data) {
8       this.data = data;
9     })
10  })
11  it('login with valid credentials', function () {
12    LoginPage.login('testautomation@cypress-test.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  })

```

- Formatting throughout project: one consistent formatting should be used.

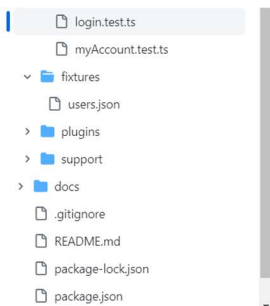


```

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 })
28 it('login with invalid password credentials read data from fixture', function () {
29   LoginPage.login(this.data.invalid_credentials.invalid_password.emailId,
30     this.data.invalid_credentials.invalid_password.password)
31   LoginPage.validateLoginError('Authentication failed.')
32 })
33 // login with wrong email format credentials read data from fixture' function () {

```

This line is too long:

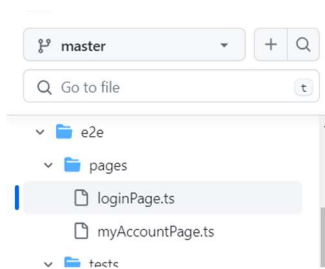


```

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 })
28 it('login with invalid password credentials read data from fixture', function () {
29   LoginPage.login(this.data.invalid_credentials.invalid_password.emailId,
30     this.data.invalid_credentials.invalid_password.password)
31   LoginPage.validateLoginError('Authentication failed.')
32 })
33 it('login with wrong email format credentials read data from fixture', function () {
34   LoginPage.login(this.data.invalid_credentials.wrong_email_format.emailId, this.data.invalid_credentials.wrong_email_format.password)
35   LoginPage.validateLoginError('Invalid email addressssss.')
36 })
37 })

```

- Put the opening bracket at the end of the first line.
- Put the closing bracket on a new line, without leading spaces.
- See https://www.w3schools.com/js/js_conventions.asp [Function section] for more information.



```

1 /// <reference types="cypress" />
2
3 class LoginPage {
4   get signInLink() { return cy.get('.login') }
5   get emailAddressTxt() { return cy.get('#email') }
6   get passwordTxt() { return cy.get('#passwd') }
7   get signInBtn() { return cy.get('#SubmitLogin') }
8   get alertBox() { return cy.get(':contains("error")') }
9   get alertMessage() { return cy.get('.alert-danger > ol > li') }
10

```

- You should stick to one coding style: either using arrow functions `() => {}` or the function keyword `function() {}`.

```

1  import { loginPage } from '../pages/loginPage'
2  import { myAccountPage } from '../pages/myAccountPage'
3
4  describe('Login Functionality', () => {
5    beforeEach(() => {
6      loginPage.launchApplication()
7      cy.fixture('users.json').then(function (data) {
8        this.data = data;
9      })
10   })
11   it('login with valid credentials', function () {
12     loginPage.login("testautomation@cypress.test.com", "Test@1234")
13     myAccountPage.validateSuccessfulLogin()
14     myAccountPage.logout()
15     myAccountPage.validateSuccessfulLogout()
16   })

```

- [MyAccount.test] Unfunctional code is present in this file. 'myAccountPage' should be imported into 'myAccount.test' file. Test cases should match the account functionality.
- [MyAccount.test] It is a bad practice to use 'console.log' in test cases.
- [MyAccount.test] Comment is inconsistent, as given url doesn't launch the app.

```

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 })

```

- It is better to remove unused, autogenerated files('commands.js').
- As 'commands.js' is totally commented there's no need to import it in 'e2e.js' file.

```

15
16 // Import commands.js using ES2015 syntax:
17 import './commands'
18 import '@shelex/cypress-allure-plugin'
19
20 // Alternatively you can use CommonJS syntax:
21 // require('./commands')

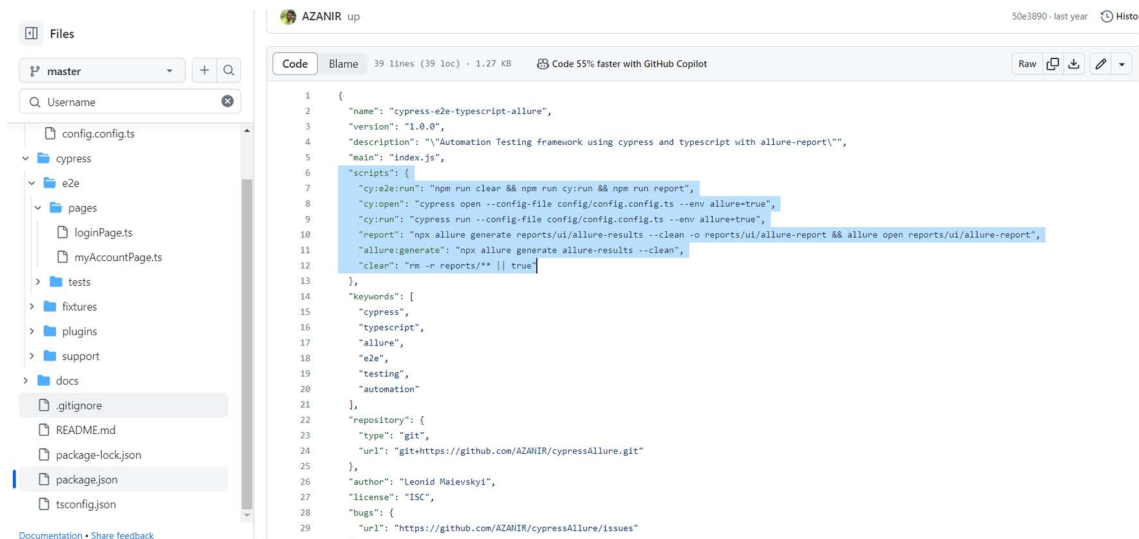
```

- 'Docs' folder seems useless. Please add it to '.gitignore'.
- 'README.md' doesn't include any information about account tests.
- Unclear step 'Change 'Username' and 'Password' in 'README.md'. No config file was mentioned.

Steps to run

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

- Not all commands are present in 'README.md'.



- [Users.json] To properly check for the 'wrong email format': 'emailId' should be intentionally in the wrong format and the password should be valid. Password value is incorrect in 'wrong_email_format' object. It has to be taken from 'valid_credentials' object.

