

```

cypress > e2e > pages > TS myAccountPage.ts > MyAccountPage > validateSuccessfulLogout
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
13     public logout() {
14         this.signoutLink.click()
15     }
16
17     public validateSuccessfulLogout() {
18         loginPage.signinLink.should('be.visible')
19     }
20 }
21
22 export const myAccountPage: MyAccountPage = new MyAccountPage()

```

e2e/pages/myAccountpage.ts

Line 18: Page object for the “My account” page uses logic from another page object class. A page object class should contain logic and locators only for one specific page. Any logic/locators that are shared between page objects can be moved to a parent class that all page object classes inherit from, usually named BasePage.

```

cypress > e2e > tests > TS myAccount.test.ts > ...
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 })

```

e2e/tests/myAccount.test.ts

Line 5: A hardcoded link is used instead of an existing method that utilizes baseUrl

Line 9: console log is unrelated to test logic

Suggestion: Any comments or code added for debugging purposes should be removed after issues are resolved

```
cypress > e2e > tests > TS login.test.ts > describe('Login Functionality') callback > it('login with valid credentials') callback
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@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 })
```

e2e/tests/login.test.ts

Line 12: Hardcoded login data in the login method call. A dedicated json file for the login data already exists and should be utilized like in the rest of the tests (for example, line 18)

```
cypress > e2e > tests > TS login.test.ts > ...
4  describe('Login Functionality', () => {
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_
35     loginPage.validateLoginError('Invalid email addressssss.')
36   })
37 })
```

Line 34: Splitting long method calls into several lines is a good idea for readability. However, the formatting should stay consistent throughout the code. The login method call on line 34 should be split the same way as on lines 24-25 and 29-30

```
cypress > plugins > JS index.js > ...
1  /// <reference types="@shelex/cypress-allure-plugin" />
2
3  const allureWriter = require('@shelex/cypress-allure-plugin/writer');
4  module.exports = (on, config) => {
5    |    allureWriter(on, config);
6    |    return config;
7  };
```

plugins/index.js

Suggestion: If possible, it's a good practice to avoid using both CommonJS and ES modules in the same project, to rule out any potential issues and to keep the code consistent.

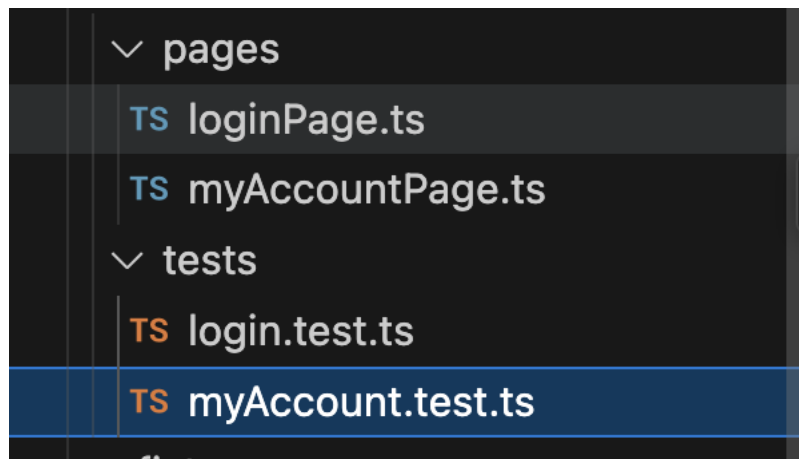
```

cypress > e2e > pages > TS loginPage.ts > LoginPage > launchApplication
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) {
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: LoginPage = new LoginPage()

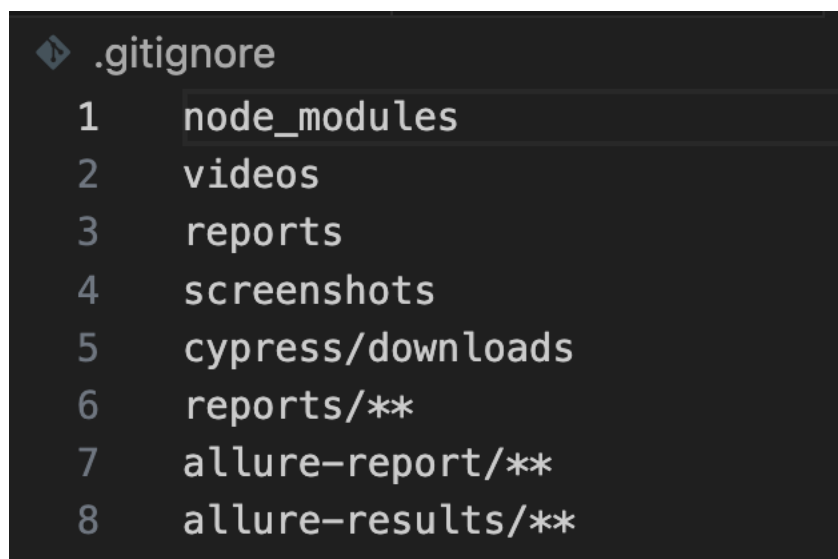
```

e2e/pages /loginPage.ts

Suggestion: Avoid adding unnecessary spaces, like on line 25. It is a good practice to use an Auto-Formatting command of your IDE (Alt + Shift + F in VSCode) on files you have worked with before making a commit.



File naming: Generally, files should be named in all-lowercase. Files that contain a class should be named the same as the class (for example, LoginPage.ts)



Suggestion: Gitignore should also contain IDE-related folders, as they could potentially cause unwanted files to be committed.