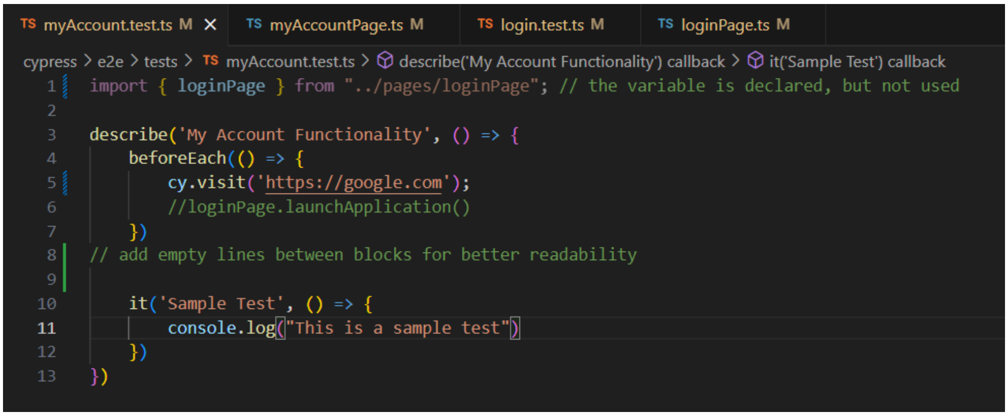
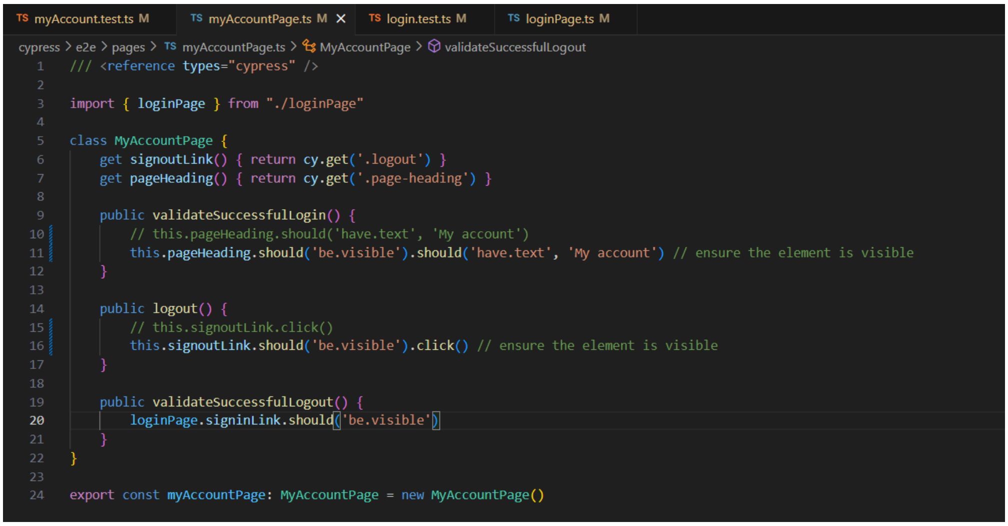
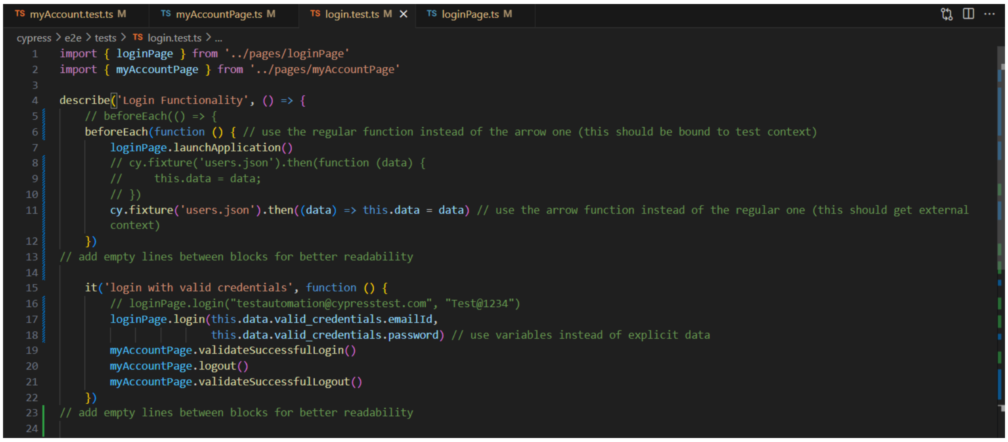

TS myAccount.test.ts M ×
TS myAccountPage.ts M
TS login.test.ts M
TS loginPage.ts M
cypress > e2e > tests > TS myAccount.test.ts > describe('My Account Functionality') callback > it('Sample Test') callback

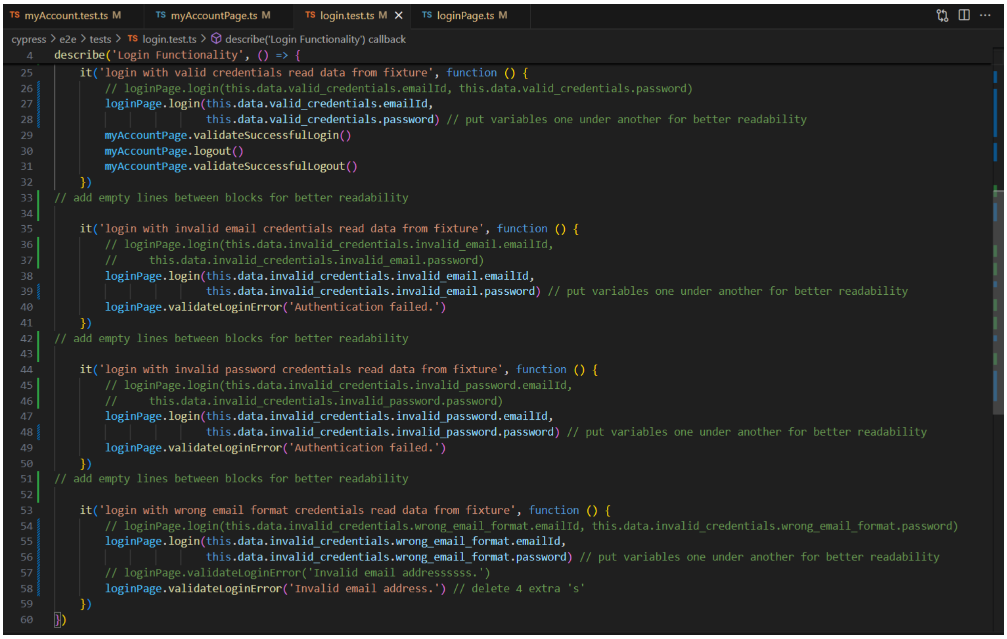
```typescript
import { loginPage } from "../pages/loginPage"; // the variable is declared, but not used

describe('My Account Functionality', () => {
    beforeEach(() => {
        cy.visit('https://google.com');
        //loginPage.launchApplication()
    })
// add empty lines between blocks for better readability

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


TS myAccount.test.ts M
TS myAccountPage.ts M ×
TS login.test.ts M
TS loginPage.ts M
cypress > e2e > pages > TS myAccountPage.ts > MyAccountPage > validateSuccessfulLogout

```typescript
/// <reference types="cypress" />

import { loginPage } from "./loginPage"

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

    public validateSuccessfulLogin() {
        // this.pageHeading.should('have.text', 'My account')
        this.pageHeading.should('be.visible').should('have.text', 'My account') // ensure the element is visible
    }

    public logout() {
        // this.signoutLink.click()
        this.signoutLink.should('be.visible').click() // ensure the element is visible
    }

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

export const myAccountPage: MyAccountPage = new MyAccountPage()
```

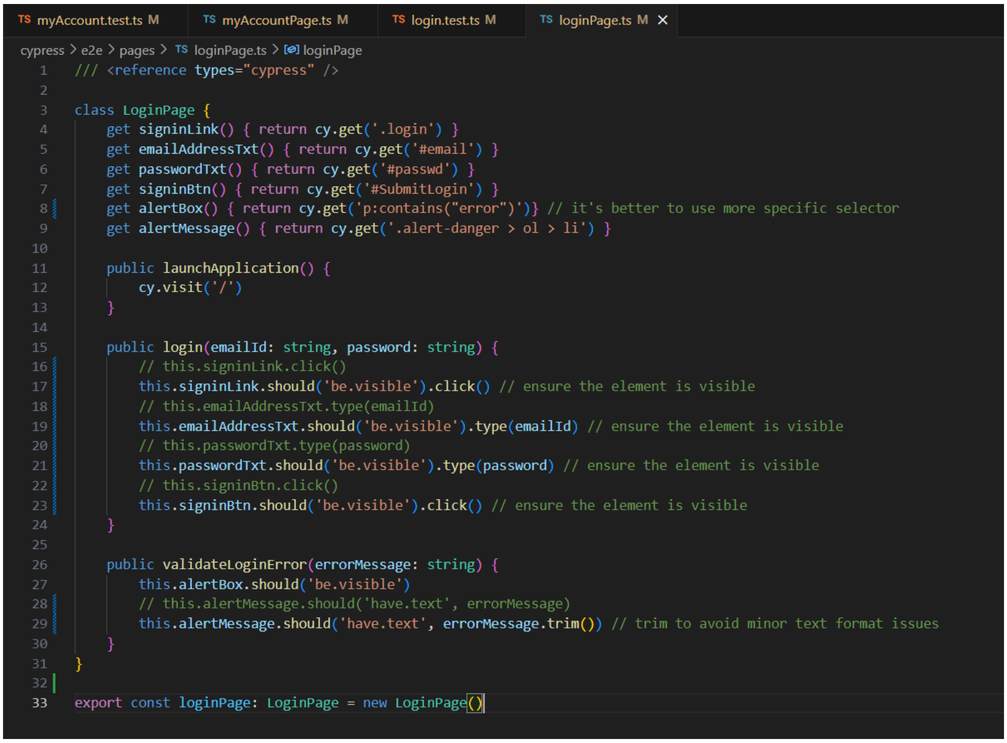


```typescript
import { loginPage } from '../pages/loginPage'
import { myAccountPage } from '../pages/myAccountPage'

describe('Login Functionality', () => {
    // beforeEach(() => {
    beforeEach(function () { // use the regular function instead of the arrow one (this should be bound to test context)
        loginPage.launchApplication()
        // cy.fixture('users.json').then(function (data) {
        //     this.data = data;
        // })
        cy.fixture('users.json').then((data) => this.data = data) // use the arrow function instead of the regular one (this should get external context)
    })
// add empty lines between blocks for better readability

    it('login with valid credentials', function () {
        // loginPage.login("testautomation@cypresstest.com", "Test@1234")
        loginPage.login(this.data.valid_credentials.emailId,
                        this.data.valid_credentials.password) // use variables instead of explicit data
        myAccountPage.validateSuccessfulLogin()
        myAccountPage.logout()
        myAccountPage.validateSuccessfulLogout()
    })
// add empty lines between blocks for better readability

```



TS myAccount.test.ts M
TS myAccountPage.ts M
TS login.test.ts M
TS loginPage.ts M
cypress > e2e > tests > TS login.test.ts > describe('Login Functionality') callback

```typescript
describe('Login Functionality', () => {
    it('login with valid credentials read data from fixture', function () {
        // loginPage.login(this.data.valid_credentials.emailId, this.data.valid_credentials.password)
        loginPage.login(this.data.valid_credentials.emailId,
                        this.data.valid_credentials.password) // put variables one under another for better readability
        myAccountPage.validateSuccessfulLogin()
        myAccountPage.logout()
        myAccountPage.validateSuccessfulLogout()
    })
// add empty lines between blocks for better readability

    it('login with invalid email credentials read data from fixture', function () {
        // loginPage.login(this.data.invalid_credentials.invalid_email.emailId,
        //     this.data.invalid_credentials.invalid_email.password)
        loginPage.login(this.data.invalid_credentials.invalid_email.emailId,
                        this.data.invalid_credentials.invalid_email.password) // put variables one under another for better readability
        loginPage.validateLoginError('Authentication failed.')
    })
// add empty lines between blocks for better readability

    it('login with invalid password credentials read data from fixture', function () {
        // loginPage.login(this.data.invalid_credentials.invalid_password.emailId,
        //     this.data.invalid_credentials.invalid_password.password)
        loginPage.login(this.data.invalid_credentials.invalid_password.emailId,
                        this.data.invalid_credentials.invalid_password.password) // put variables one under another for better readability
        loginPage.validateLoginError('Authentication failed.')
    })
// add empty lines between blocks for better readability

    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.login(this.data.invalid_credentials.wrong_email_format.emailId,
                        this.data.invalid_credentials.wrong_email_format.password) // put variables one under another for better readability
        // loginPage.validateLoginError('Invalid email addressssss.')
        loginPage.validateLoginError('Invalid email address.') // delete 4 extra 's'
    })
})
```



```typescript
/// <reference types="cypress" />

class LoginPage {
    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")')} // it's better to use more specific selector
    get alertMessage() { return cy.get('.alert-danger > ol > li') }

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

    public login(emailId: string, password: string) {
        // this.signinLink.click()
        this.signinLink.should('be.visible').click() // ensure the element is visible
        // this.emailAddressTxt.type(emailId)
        this.emailAddressTxt.should('be.visible').type(emailId) // ensure the element is visible
        // this.passwordTxt.type(password)
        this.passwordTxt.should('be.visible').type(password) // ensure the element is visible
        // this.signinBtn.click()
        this.signinBtn.should('be.visible').click() // ensure the element is visible
    }

    public validateLoginError(errorMessage: string) {
        this.alertBox.should('be.visible')
        // this.alertMessage.should('have.text', errorMessage)
        this.alertMessage.should('have.text', errorMessage.trim()) // trim to avoid minor text format issues
    }
}

export const loginPage: LoginPage = new LoginPage()
```