

cypressAllure

- ❖ В tsconfig.json не вказаний baseUrl
- ❖ Readme:
 - Використання заголовків де не потрібно

cypress-e2e-allure# Cypress-e2e-Allure Automation-Framework with allure reports

This Framework contains sample code for # Cypress-e2e-Allure Automation-Framework

- Page Object Model
 - Reading test data from json file
 - Simple login scenario with valid credentials and with invalid credentials
 - Allure report integration
- Мало інформації про запуск (відсутні кроки створення репорту)
 - Незрозумілі кроки запуску

Steps to run

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

Де саме змінити username і password?

- ❖ В package.json є необхідні скрипти (наприклад, для створення репорту), однак в Readme про них жодної інформації.
- ❖ В .gitignore внесені не всі необхідні папки

cypressAllure\.github\workflows

- ❖ В cyAllureRun.yml присутній коментований код

```
runs-on: ubuntu-20.04 # Choosing OS
strategy:
  matrix:
    node-version: [14.17.0]
    # See supported Node.js release schedule at https://nodejs.org/en/about/releases/
steps:
  - uses: actions/checkout@v2
  - name: Use Node.js ${{ matrix.node-version }}
    uses: actions/setup-node@v2
    with:
      node-version: ${{ matrix.node-version }}
      cache: "npm"
  - run: npm i npm@latest
  - run: npm i
  - name: run test
    run: npm run cy:run
  - name: Create Report
    if: always()
    run: npm run allure:generate
  - name: Upload artifact allure-report
    uses: actions/upload-artifact@v1
    if: always()
    with:
      name: allure-report
      path: ./allure-report
  - name: Deploy report
    uses: JamesIves/github-pages-deploy-action@3.1.0
    if: always()
    with:
      ACCESS_TOKEN: ${{ 'secrets.ACCESS_TOKEN' }}
      BRANCH: master # The branch the action should deploy to.
      FOLDER: allure-report # The folder the action should deploy.
      TARGET_FOLDER: docs
```

Не впевнений, чи воркфлоу працює сам по собі, оскільки версія застаріла

cypressAllure\config

- ❖ В файлі config.config.ts присутній коментований код

```
videosFolder: 'reports/videos',  
env: {  
  allureResultsPath: '../allure-results',  
},  
e2e: {  
  // We've imported your old cypress plugins here.  
  // You may want to clean this up later by importing these.
```

cypressAllure\docs

- ❖ Не впевнений, чи цю папку взагалі слід пушити до репозиторію, оскільки це інформація про репорти та налаштування Allure.
- ❖ Ці репорти мають генеруватися або локально, або у workflow і деплоїтись, наприклад, на Github Pages. Але в робочому репозиторії їх бути не має.

cypressAllure\cypress\fixtures

OK

cypressAllure\cypress\plugins

OK

cypressAllure\cypress\support

❖ В commands.js є коментований код

```
> .github
> config
  > cypress
    > e2e
      > pages
      > tests
      > fixtures
      {} users.json
    > plugins
      {} index.js
    > support
      {} commands.js
      {} e2e.js
    > docs
    > .gitignore
    {} package-lock.json
    {} package.json
    {} README.md
    {} tsconfig.json 2

1 // *****
2 // This example commands.js shows you how to
3 // create various custom commands and overwrite
4 // existing commands.
5 //
6 // For more comprehensive examples of custom
7 // commands please read more here:
8 // https://on.cypress.io/custom-commands
9 // *****
10 //
11 //
12 // -- This is a parent command --
13 // Cypress.Commands.add('login', (email, password) => { ... })
14 //
15 //
16 // -- This is a child command --
17 // Cypress.Commands.add('drag', { prevSubject: 'element'}, (subject, options) => { ... })
18 //
19 //
20 // -- This is a dual command --
21 // Cypress.Commands.add('dismiss', { prevSubject: 'optional'}, (subject, options) => { ... })
22 //
23 //
24 // -- This will overwrite an existing command --
25 // Cypress.Commands.overwrite('visit', (originalFn, url, options) => { ... })
26
```

❖ В e2e.js є коментований код

```
✓ CYPRESSALLURE
> .github
> config
  > cypress
    > e2e
      > pages
      > tests
      > fixtures
      {} users.json
    > plugins
      {} index.js
    > support
      {} commands.js
      {} e2e.js
    > docs
    > .gitignore
    {} package-lock.json
    {} package.json
    {} README.md
    {} tsconfig.json 2

cypress > support > {} e2e.js
1 // *****
2 // This example support/index.js is processed and
3 // loaded automatically before your test files.
4 //
5 // This is a great place to put global configuration and
6 // behavior that modifies Cypress.
7 //
8 // You can change the location of this file or turn off
9 // automatically serving support files with the
10 // 'supportFile' configuration option.
11 //
12 // You can read more here:
13 // https://on.cypress.io/configuration
14 // *****
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')
22
```

cypressAllure\cypress\e2e\pages

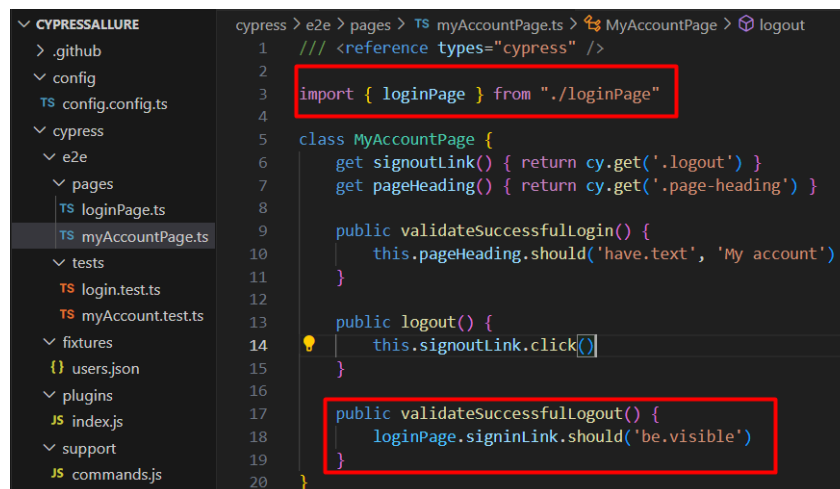
- ❖ В loginPage.ts присутній зайвий ентер



- ❖ В loginPage.ts, можливо, неправильний локатор (без запуску тестів сказати важко)

```
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")') }  
  get alertMessage() { return cy.get('.alert-danger > ol > li') }  
}
```

- ❖ В myAccountPage.ts присутній метод, що мав би належати сторінці loginPage. Використовується імпорт сторінки loginPage і жодних елементів myAccountPage. Тож слід прибрати import і перемістити метод validateSuccessfulLogout в файл loginPage.ts



cypressAllure\cypress\e2e\tests

❖ login.test.ts

- В присутні два тести, що виконують одну й ту ж дію. Для введення валідних даних слід використовувати json файли з fixtures, тож тест “login with valid credentials” можна виключити зі сьюта.

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

- В в трьох тестах для використання невалідних даних використовуються сталі значення з json файлу. Натомість слід використовувати генератор випадкових значень (наприклад, faker)

```
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.validateLoginError('Authentication failed.')
})
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.validateLoginError('Authentication failed.')
})
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.')
})
```

- В цих же тестах для валідації повідомлення про помилку можна було б використовувати json файл з fixtures, оскільки деякі значення повторюються

```
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.validateLoginError('Authentication failed.')
})
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.validateLoginError('Authentication failed.')
})
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.')
})
```

❖ myAccount.test.ts

- В коді присутній імпорт, що не використовується

```
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  })
```

- Присутній коментований код

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

- В тесті присутнє виведення тексту в консоль

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

- В тесті нема жодного expect, assert або should

Загалом виглядає так, наче цей спрес потрібен для перевірки, чи запускається сайт загалом і чи є можливість запускати тести. Тож його не слід було пушити в репозиторій з проектом. Або ж додати перевірку іншим чином.