1. Using always the same data for tests is a bad practice, which can lead to the well-known [Pesticide Paradox] (system may become immune to one or another unpredictable interaction).

   You have hardcoded data in file users.json

   So it's better to use random generated data for test cases such as registrations etc. You can make your own methods for this generation or you can use faker-js

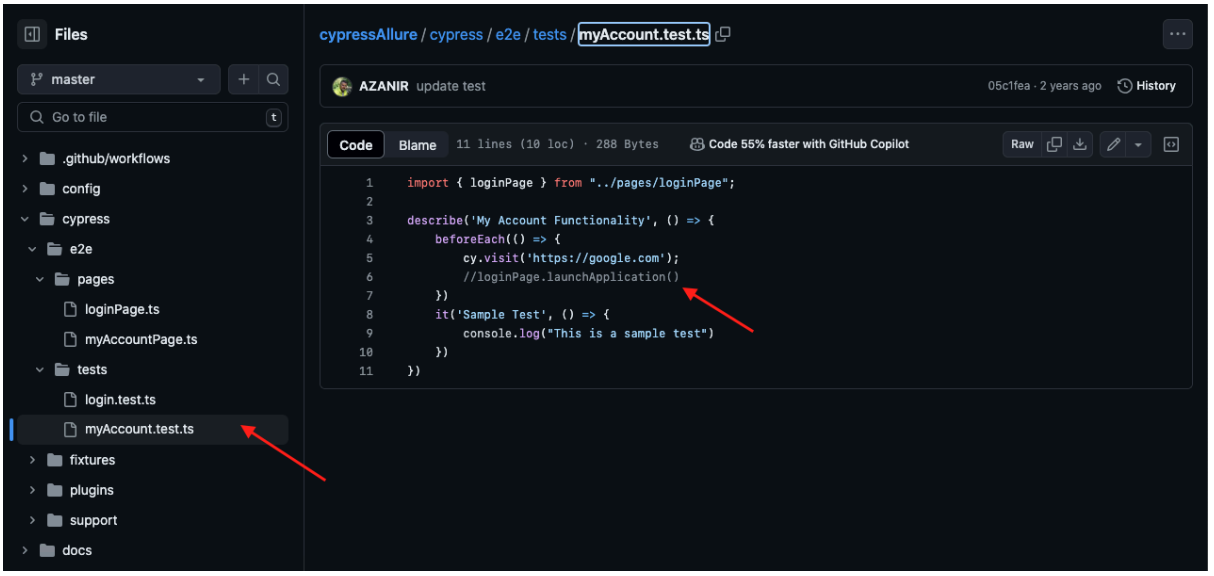
2. Repo has many redundant files that were created by framework or reporter

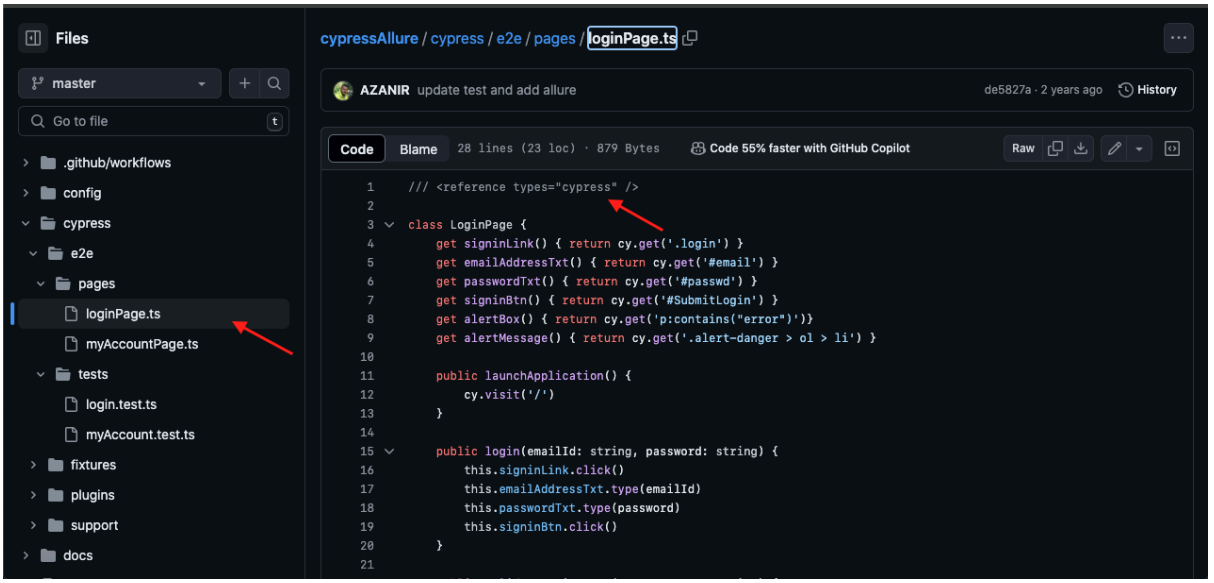
   You have folder "docs" which was created by reporter, it's better to add folder to gitignore file

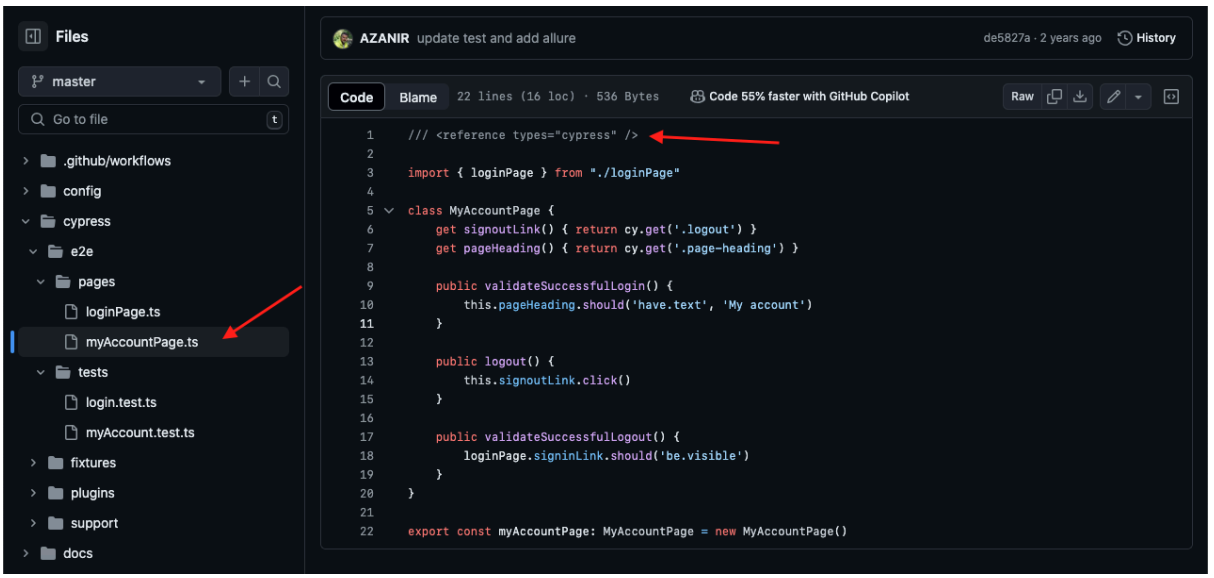
   And you have

3. It's a good practice to name your files in the same pattern :
   for example you have login.test.ts file for spec and loginPage.ts file for POM, so it's better to name files with the same pattern : loginPage.ts and loginPage.ts, or login.test.ts and login.page.ts
   And the same with "my account" files

4. Code is't formatted
   The repository should be as readable as possible, delete all the unnecessary commented part of the code.







5. myAccountPage.ts file is't completed

   It looks like it's a sample and it doesn't include specs for account functionality of testing website. Every test case should include at least one assertion instead of console log. The output to the console will not stop the execution of the test, and in the results will not indicate the error that was detected as a result of testing.

   You import class LoginPage and don't use it. It's better to delete all the unnecessary imports

   Better to validate by URL instead of

   public validateSuccessfulLogin() {

```
        this.pageHeading.should('have.text', 'My account')
    }
```

6. Recommendations for login.test.ts

   Loading fixture data inside beforeEach might cause issues due to asynchronous nature. Consider loading it once in a higher scope if it doesn't change between tests.

   Consider using arrow functions for callbacks in describe and it for better readability.

   You have repeated code for logging in, logging out, and validating login and logout. Consider creating reusable functions for these actions.

   Use data from user.json instead of hardcoded data in “Login with valid credentials” test case

7. Recommendations for git hub actions :

   Combine npm steps for efficiency and use ci Instead of i for npm Install:
   run: npm install -g npm@latest && npm ci

   Use the latest versions of actions where possible:
   - uses: actions/upload-artifact@v3

   The ACCESS_TOKEN should be referenced correctly without quotes