Code review of the cypressAllure project.

GitHub link: https://github.com/AZANIR/cypressAllure.git

1. **Locators:**
   Locators are initialized as getters. Some selectors may not work in different cases:
   get alertBox() { return cy.get('p:contains("error")')}

   Recommendation: Locators should be initialized as a method in the elements object of the class for more flexibility and dynamic retrieval of elements. Example:

   ```
   class homePage{
      elements ={
         loginBtn : () => cy.get(“#signin”),
         logOffBtn : () => cy.get(“#logout”)
      }

      clickOnSignin(){
         this.elements.loginBtn().click()
      }
   }

   module.exports = new homePage();
   ```

   Materials: https://www.browserstack.com/guide/cypress-page-object-model
   Check selectors for reliability.
   Materials:
   https://www.browserstack.com/guide/find-html-element-using-cypress-locators

2. **Testing data:**
   Test data is hard-coded in the .json file.

   Recommendation: Use some tools to generate random data to avoid problems with testing on static data. Using static data in some test cases like "Login with valid data" is possible, but it should be stored as a constant in the test.

3. **Verification:**
   Some tests use 'console.log' instead of 'expect' and hard-coded values.
   For example:

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

   Recommendations: Ensure that all important checks are performed with 'expect'. Verification should be re-usable if possible.

4. **POM:**
   Cypress commands are used inside the test file instead of inside the class method. For example:

```
beforeEach(() => {
    cy.visit('https://google.com');
    //loginPage.launchApplication()
})
```

   Class myAccountPage includes checking from another Page Object class:

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

   Recommendation: Cypress commands should be used inside methods of a Page Object class for more flexibility. Use methods from the same Page Object class for checks.

5. **Allure-commandline:**
   Allure-commandline is installed globally.

   Recommendation: Install allure-commandline locally and use it via npx in your scripts:
   npm install allure-commandline --save-dev - install locally

6. **Configuration:**
   It looks like config files contain duplicate or unnecessary settings and incorrect paths. For example tsconfig.json:

```
"exclude": ["node_modules"],
"include": ["**/*.ts",
  "../node_modules/@shelex/cypress-allure-plugin/reporter",
  "../node_modules/cypress"]
```

   Recommendation: Make sure that the configuration is up-to-date and free of duplicates. Check paths to files and directories.

7. **Code:**
   The code is not formatted and contains unnecessary comments.

   Recommendation: The repository should be as readable as possible, it should not contain any extra spaces or entries, commented part of the code. To format the file, it is enough to remove all unnecessary entries and code and then press Shift+Alt+F (in VS Code).

8. **Project Structure:**
   The repository contains redundant directories(eg 'docs' directory).

   Recommendation: Exclude unnecessary directories.

9. Project description:

The URL of the repo does not exist in the step to clone the project. The second step in the "Steps to run" section is unnecessary. Description for commands does not exist.

Recommendation: The description of the project should provide an overview with information about the project, steps to install, steps to launch, steps to generate a report, etc. Steps or sections should contain an additional description of commands or examples of code.

10. GitHub Actions:

The '.github/workflow' directory exists but lacks '.yml' files.

Recommendation: Set up GitHub Actions (pipeline) for continuous integration
Materials: https://github.com/cypress-io/github-action.git