Code review for
https://github.com/AZANIR/cypressAllure/tree/master/cypress

Strengths:

1. **Page Object Model (POM):** The project employs POM, which enhances test maintenance and readability by encapsulating page-specific information.

2. **Allure Report Integration:** The integration with Allure provides detailed and visually appealing test reports, aiding in result analysis.

3. **Test Data Management:** Utilizing JSON files for test data promotes separation of data from test scripts, facilitating easier data management.

Areas for Improvement:

1. **Documentation Enhancement:**

    - **Setup Instructions:** The README provides basic steps to run the project but lacks detailed setup instructions. Including prerequisites, environment configurations, and potential troubleshooting tips would benefit new users.

    - **Project Structure Explanation:** While there's an image depicting the project structure, a textual explanation would improve accessibility and understanding.

2. **Handling Asynchronous Behavior:**

    - Cypress commands are asynchronous, and neglecting this can lead to flaky tests. Ensuring proper handling of asynchronous operations, such as waiting for elements to appear before interacting, is vital.

3. **Configuration Management:**

    - Setting a global `baseUrl` in the Cypress configuration can simplify test scripts by allowing relative URLs in `cy.visit()` commands.

4. **Version Compatibility:**
    - Ensuring that all dependencies, especially `cypress` and `cypress-allure-plugin`, are compatible is crucial. Mismatched versions can lead to integration issues.

5. **Data handling**
    - Sensitive data, such as usernames and passwords, are mentioned in the README.md as needing changes. It's advisable to avoid hardcoding sensitive information.
6. Scripts cy:open and cy:run should be run via npx to avoid running cypress binary directly
7. beforeEach(() => {
   cy.visit('https://google.com');
   //loginPage.launchApplication()
   })
   Why visit google.com when you are testing My Account functionality
   Suggestion:
   beforeEach(() => {
   loginPage.launchApplication();
   });
8. console.log("This is a sample test")
   console.log doesnt show output in cypress ui, it`s better to use cy.log() for this scenario
9. validateLoginError('Invalid email addresssssss.') - typo
10. Selector for alertBox might be too loose
    get alertBox() { return cy.get('p:contains("error")') }

Recommendations:

Expand Documentation: Enhance the README with detailed setup instructions, including prerequisites (Node.js version, Cypress install method), configuration steps, and common troubleshooting tips. A brief description of each folder/file in the project structure will also help new contributors navigate the codebase more easily.

Improve Test Consistency: Replace placeholder or irrelevant navigation like `cy.visit('https://google.com')` with actual app entry points, such as calling `loginPage.launchApplication()` to reflect real user flows and ensure test reliability.

Handle Asynchronous Behavior Properly: Make sure commands that depend on DOM changes are preceded by appropriate waits or assertions. Use `cy.should()` or `cy.contains()` when waiting for elements rather than relying on timing

assumptions.

Use Cypress-Preferred Logging: Replace `console.log()` with `cy.log()` to ensure that logs are visible within the Cypress Test Runner UI, improving visibility during debugging.

Fix Typographical Errors and Selectors:

- Correct typos in expected error messages (e.g., `'Invalid email addressssss.'`).
- Tighten selectors to avoid flaky test behavior. For example, `cy.get('p:contains("error")')` might match unintended elements — consider using a `data-testid` attribute or a more specific class.

Centralize and Secure Test Data: Avoid suggesting manual edits of credentials in documentation. Instead, manage sensitive test data via environment variables or use `cypress.env.json` with `.gitignore` to avoid exposing credentials.

Configure `baseUrl`: Define the base URL in `cypress.config.js` to streamline test navigation. This allows you to use relative paths in `cy.visit()` and keeps tests more maintainable.

Run Cypress via `npx`: Update the test scripts to use `npx cypress open` and `npx cypress run` instead of calling Cypress directly. This ensures compatibility regardless of global installs and avoids path issues on different machines.

Verify Dependency Compatibility: Regularly check that all installed packages, particularly `cypress` and `cypress-allure-plugin`, are compatible and up to date to prevent integration or runtime errors.