Code Review

1. Instead of using the same testing data that can lead to Pesticide paradox, it would be better to use faker package for testing data generation.

```
const { faker } = require('@faker-js/faker');

const randomEmail = faker.internet.email();
const randomPassword = faker.internet.password();

it('login with invalid email credentials, function () {
    loginPage.login(randomEmail, randomPassword)
    loginPage.validateLoginError('Authentication failed.')
  })
```

2. It is unsafe to store sensitive data in the code directly. It is a good practice to save sensitive data in .env file for better data security.

To do this, you need:

   a. Install dotenv package;
   b. Create a .env file and add your credentials;
   c. Configure Cypress to use environment variables;
   d. Access environment variables in your tests using Cypress.env.

3. To make the code more readable, the redundant code should be removed.

```
import { loginPage } from "../pages/loginPage";

describe('My Account Functionality', () => {
  beforeEach(() => {
    cy.visit('https://google.com');
```

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

4. The name of test suit does not correspond to its test.

```
import { loginPage } from "../pages/loginPage";

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

5. Tests should have at least 1 assertion instead of console.log().

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

6. The testing results should be added to another branch to prevent conflicts which can appear by pushing new features.