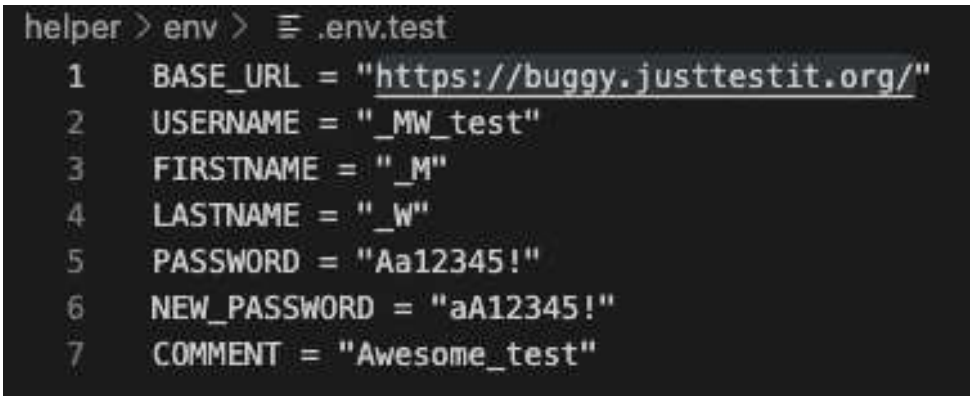
Playwright buggy rating framework user guide

1. How to add a new test case within this framework?

a. Add a child page object class which inherits from parent class Wrapper within folder /src/pages, add your config data into config file within folder helper/env/ and new variable into /src/utils/env.ts if needed.

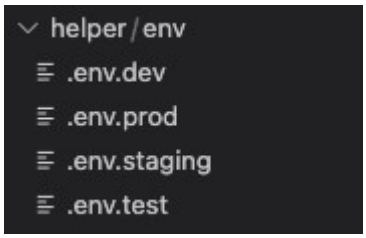


b. Add a facade class into /src/design_patterns folder, you are able to implement actual test cases cross different page objects, just passing page object parameters into constructor function.

c. Declare page object and facade variables within /src/tests/buggyrating.spc.ts, initialize variables within test.beforeAll function, add a new test case within test.describe function.

2. Multi environments running

a. Config data against different environment are stored in corresponding config file within folder /helper/env



b. Command lines to run test cases against different environment are set within “scripts” field of package.json

3. Organize test case by tags

a. You can organize test case by tags by adding tag name into test description, such as @sanity, @regression or @uat, of course, you can add multiple tags into a test case.

```
test('should login buggy rating system @sanity @regression', async () => {
    await loginTest.run();
});

test('should post a comment to the popular make @regression', async () => {
    await postCommentTest.run();
});
```

b. Only use cases with sanity tags are executed by add “--grep @sanity” into command line.

```
"scripts": {
  "env:dev:sanity": "cross-env test_env=dev npx playwright test --grep @sanity",
  "env:dev:regression": "cross-env test_env=dev npx playwright test --grep @regression",
  "env:test:sanity": "cross-env test_env=test npx playwright test --grep @sanity",
  "env:test:regression": "cross-env test_env=test npx playwright test --grep @regression",
  "env:staging:sanity": "cross-env test_env=staging npx playwright test --grep @sanity",
  "env:staging:regression": "cross-env test_env=staging npx playwright test --grep @regression",
  "env:prod:sanity": "cross-env test_env=prod npx playwright test --grep @sanity",
  "env:prod:regression": "cross-env test_env=prod npx playwright test --grep @regression"
},
```

4. Headless mode

a. You can change value of “headless” to “false” within file playwright.config.ts to shut down headless mode, which will be useful for code debugging.

```
TS playwright.config.ts > config > use
1  import { PlaywrightTestConfig } from '@playwright/test';
2
3  const config: PlaywrightTestConfig = {
4    testMatch: ["tests/buggyrating.spec.ts"],
5    timeout: 30000,
6    reporter: "html",
7    use: {
8      headless: true,
9    },
10   globalSetup: "src/utils/globalSetup.ts"
11 };
12
13 export default config;
```

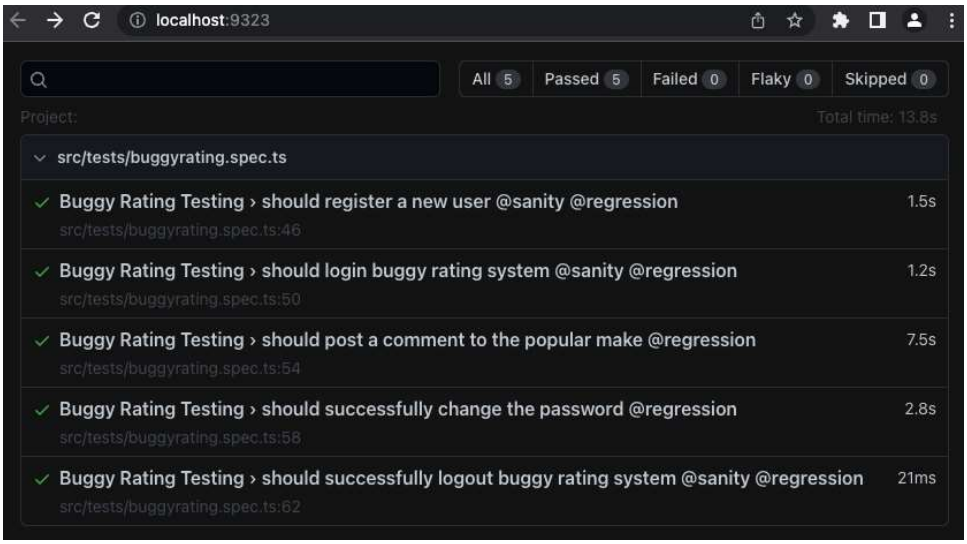
5. Design Pattern

a. I use “facade” pattern within this framework to mask complex structural code which make the code of the spec file concise and clear.

```
test('should register a new user @sanity @regression', async () => {
    await registerTest.run();
});

test('should login buggy rating system @sanity @regression', async () => {
    await loginTest.run();
});

test('should post a comment to the popular make @regression', async () => {
    await postCommentTest.run();
});

test('should successfully change the password @regression', async () => {
    await resetPasswordTest.run();
});

test('should successfully logout buggy rating system @sanity @regression', async () => {
    await logoutTest.run();
});
```

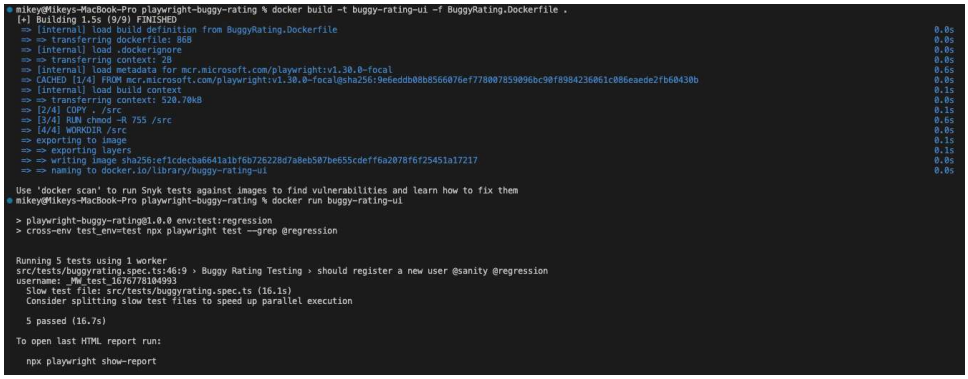
6. View report

a. It's easy to get report by running command " npx playwright show-report ", since the report is close to natural language, so it can be easily understood by non-technical stakeholders and it is no need to apply separate BDD infrastructure.



7. CI support

a. I integrate this framework with container running which make it easy to integrate framework to pipeline.



8. Username and comment data naming tips

a. I attach timestamp to the end of username and comment which make it easier to track down issues, especially within pipeline.