

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/utls/env.ts if needed.

```
helper > env > .env.test
1  BASE_URL = "https://buggy.justtestit.org/"
2  USERNAME = "_MW_test"
3  FIRSTNAME = "_M"
4  LASTNAME = "_W"
5  PASSWORD = "Aa12345!"
6  NEW_PASSWORD = "aA12345!"
7  COMMENT = "Awesome_test"

src > utls > ts env.ts > ENV
1  export default class ENV {
2    public static BASE_URL = process.env.BASE_URL !== undefined ? process.env.BASE_URL : 'https://buggy.justtestit.org/';
3    public static USERNAME = process.env.USERNAME !== undefined ? process.env.USERNAME : '_MW';
4    public static FIRSTNAME = process.env.FIRSTNAME !== undefined ? process.env.FIRSTNAME : '_M';
5    public static LASTNAME = process.env.LASTNAME !== undefined ? process.env.LASTNAME : '_W';
6    public static PASSWORD = process.env.PASSWORD !== undefined ? process.env.PASSWORD : 'Aa12345!';
7    public static NEW_PASSWORD = process.env.NEW_PASSWORD !== undefined ? process.env.NEW_PASSWORD : 'aA12345!';
8    public static COMMENT = process.env.COMMENT !== undefined ? process.env.COMMENT : 'Awesome';
9  }
```

- 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

```
▼ helper / env
  ≡ .env.dev
  ≡ .env.prod
  ≡ .env.staging
  ≡ .env.test
```

- 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 > [e] 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.

Project:	Total time: 13.8s
src/tests/buggyrating.spec.ts	
✓ Buggy Rating Testing › should register a new user @sanity @regression	1.5s
src/tests/buggyrating.spec.ts:46	
✓ Buggy Rating Testing › should login buggy rating system @sanity @regression	1.2s
src/tests/buggyrating.spec.ts:50	
✓ Buggy Rating Testing › should post a comment to the popular make @regression	7.5s
src/tests/buggyrating.spec.ts:54	
✓ Buggy Rating Testing › should successfully change the password @regression	2.8s
src/tests/buggyrating.spec.ts:58	
✓ Buggy Rating Testing › should successfully logout buggy rating system @sanity @regression	21ms
src/tests/buggyrating.spec.ts:62	

7. CI support

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

```

mike@Mike-MacBook-Pro: ~/playwright-buggy-rating % docker build -t buggy-rating-ui -f BuggyRating.Dockerfile .
[+] Building 1.5s (9/9) FINISHED
=> [internal] load build definition from BuggyRating.Dockerfile
=> transferring dockerfile: 86B
=> [internal] load .dockerignore
=> transferring context: 2B
=> [internal] load metadata for mcr.microsoft.com/playwright:v1.30.0-focal
=> CACHED [1/4] FROM mcr.microsoft.com/playwright:v1.30.0-focal@sha256:9e6edd00b8566076e778007259896bc98789842360c1c886eade27b60430b
=> [internal] load build context
=> transferring context: 520.7kB
=> [2/4] COPY . /src
=> [3/4] RUN chmod -R 755 /src
=> [4/4] WORKDIR /src
=> exporting to image
=> exporting layers
=> writing image sha256:ef1cdecba6641a1bf6b726228d7a8eb507be655cdef6a2078f6f25451a17217
=> naming to docker.io/library/buggy-rating-ui

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
mike@Mike-MacBook-Pro: ~/playwright-buggy-rating % docker run buggy-rating-ui
> playwright-buggy-rating@1.0.0 env:test:regression
> cross-env test_env=test npx playwright test --grep @regression

Running 5 tests using 1 worker
src/tests/buggyrating.spec.ts:46:9 › Buggy Rating Testing › should register a new user @sanity @regression
username: 'm_test' (16.7718499s)
Slow test file: src/tests/buggyrating.spec.ts (16.1s)
Consider splitting slow test files to speed up parallel execution
5 passed (16.7s)

To open last HTML report run:
npx playwright show-report

```

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.