I) Introduction

When Felix indicated to me that the Lithia Driveway team was interested in someone with Playwright experience, I thought it would be a good idea to create a simple Playwright demo project using the Driveway URL.

The key weakness of Gui testing is unnecessary flakiness. My intent is to show how I supplement/work with Playwright with reliability-first practices so that the Gui tests are as reliable as possible, along with faster debugging and maintenance response time.

While an SDET should not be evaluated based on one testing framework, I hope the project achieves its purpose above.

Also it is important to point out that a Gui testing framework needs to be paired and working with a (more reliable/faster maintenance) API testing framework in order to deal with the overall goal of fast testing and thorough verification of the website/product.

Benign testing

I am using my own login account/password to benignly test it on www.driveway.com. By benign, I mean to judiciously use the website and not cause a traffic or incursion incident.

This generally means that when I run a loop test, the loop waits 1 hour between each test.

Areas not addressed

There are a number of areas of improvements that I have not had time to address such as :

1. Better documentation of the code

2. Description of the uses of test folders : unit, qe, prod etc

II) How to run

1. Unzip the distribution

2. Update the entries for a valid userEmail, userPassword and userName in test file playwright/tests/unit/unit.e2e-spec.ts

3. Install Playwright with

   a. npm ci

   b. npx playwright install

4. Run the Chromium test with 'npm run test:debug'

III) Brief summary of results

The first implementation of login (function 'login') appeared to work on my machine but its NumToFail loop ranged from 2 to 10, for Chromium.

When I ran the same tests on my Ubuntu machine (on WSL2), the failures become more obvious. It seems that my Ubuntu machine is somewhat slower and this helps expose the timing problems.

The improved implementation of login (function 'login_qe') shows that the page switches between several pages during the login process. The stable end point is when the page url becomes 'https://www.driveway.com/mydriveway'.

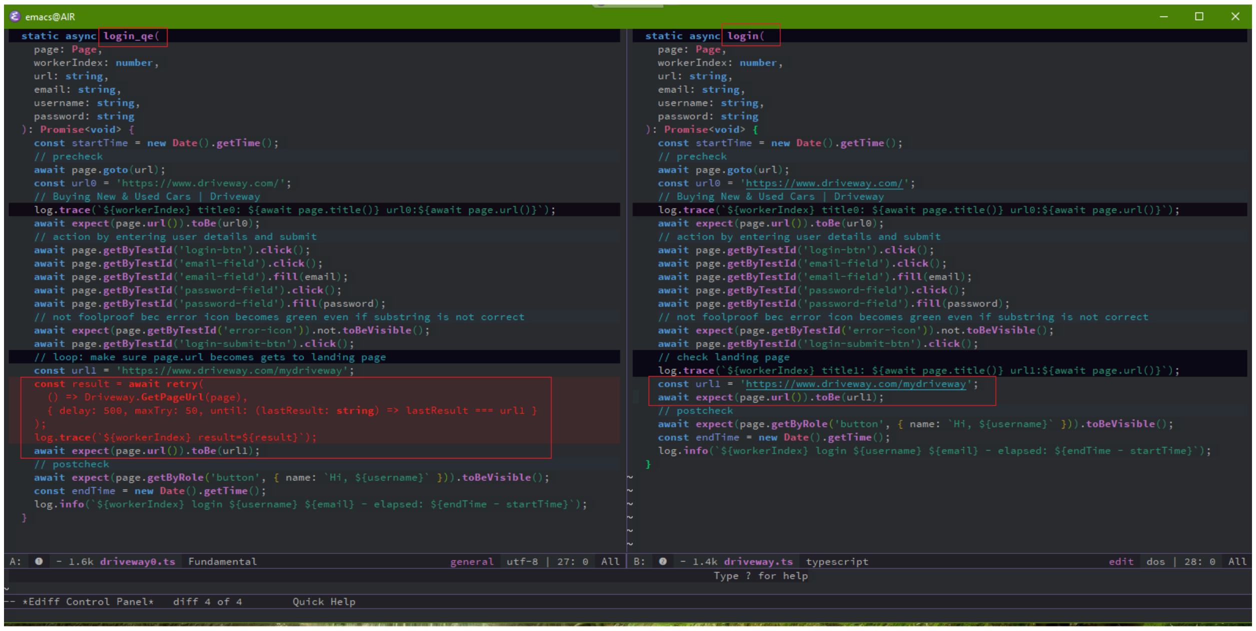
IV) Compare initial 'login' implementation with improved 'login_qe'

Dealing with flaky Gui tests

Flaky tests is not just due to poor coding practice. There are several areas of concern that helps to deal with them.

From the coding point of view, the main idea is to isolate all user actions into pure (as much as possible) functions that can be tested in many different scenarios such as unit testing, loop/soak testing and finally production testing.

Loop/soak testing is key to helping surface timing issues so that the test can deal with them early. Also, adding checks all over the code will just slow the test down, so the loop tests identify the key places that needs the checks and not every line of code.



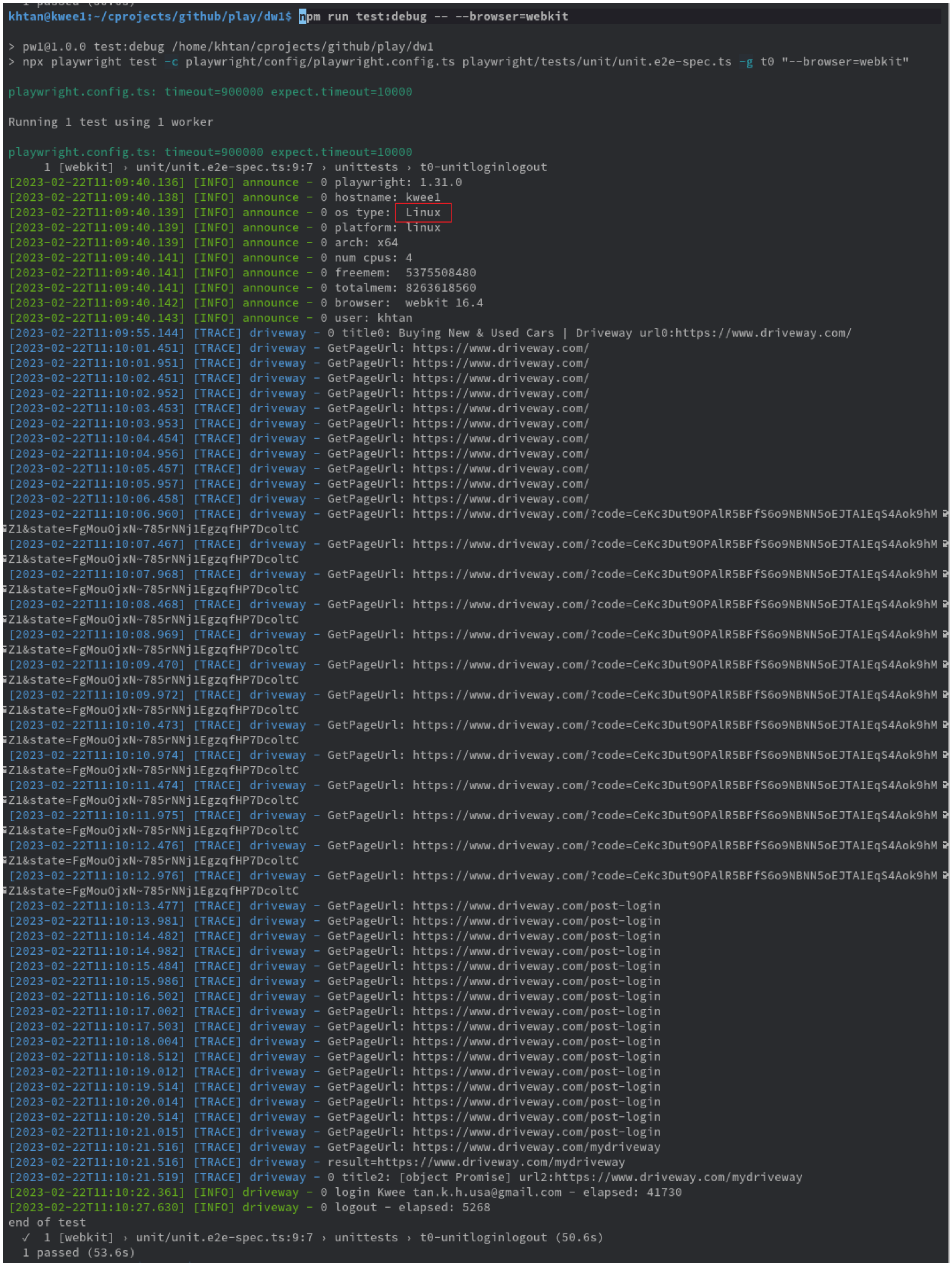
```
static async login_qe(
  page: Page,
  workerIndex: number,
  url: string,
  email: string,
  username: string,
  password: string
): Promise<void> {
  const startTime = new Date().getTime();
  // precheck
  await page.goto(url);
  const url0 = 'https://www.driveway.com/';
  // Buying New & Used Cars | Driveway
  log.trace(`${workerIndex} title0: ${await page.title()} url0:${await page.url()}`);
  await expect(page.url()).toBe(url0);
  // action by entering user details and submit
  await page.getByTestId('login-btn').click();
  await page.getByTestId('email-field').click();
  await page.getByTestId('email-field').fill(email);
  await page.getByTestId('password-field').click();
  await page.getByTestId('password-field').fill(password);
  // not foolproof bec error icon becomes green even if substring is not correct
  await expect(page.getByTestId('error-icon')).not.toBeVisible();
  await page.getByTestId('login-submit-btn').click();
  // loop: make sure page.url becomes gets to landing page
  const url1 = 'https://www.driveway.com/mydriveway';
  const result = await retry(
    () => Driveway.GetPageUrl(page),
    { delay: 500, maxTry: 50, until: (lastResult: string) => lastResult === url1 }
  );
  log.trace(`${workerIndex} result=${result}`);
  await expect(page.url()).toBe(url1);
  // postcheck
  await expect(page.getByRole('button', { name: `Hi, ${username}` })).toBeVisible();
  const endTime = new Date().getTime();
  log.info(`${workerIndex} login ${username} ${email} - elapsed: ${endTime - startTime}`);
}

A:  - 1.6k driveway0.ts  Fundamental          general  utf-8 | 27: 0  All
```

```
static async login(
  page: Page,
  workerIndex: number,
  url: string,
  email: string,
  username: string,
  password: string
): Promise<void> {
  const startTime = new Date().getTime();
  // precheck
  await page.goto(url);
  const url0 = 'https://www.driveway.com/';
  // Buying New & Used Cars | Driveway
  log.trace(`${workerIndex} title0: ${await page.title()} url0:${await page.url()}`);
  await expect(page.url()).toBe(url0);
  // action by entering user details and submit
  await page.getByTestId('login-btn').click();
  await page.getByTestId('email-field').click();
  await page.getByTestId('email-field').fill(email);
  await page.getByTestId('password-field').click();
  await page.getByTestId('password-field').fill(password);
  // not foolproof bec error icon becomes green even if substring is not correct
  await expect(page.getByTestId('error-icon')).not.toBeVisible();
  await page.getByTestId('login-submit-btn').click();
  // check landing page
  log.trace(`${workerIndex} title1: ${await page.title()} url1:${await page.url()}`);
  const url1 = 'https://www.driveway.com/mydriveway';
  await expect(page.url()).toBe(url1);
  // postcheck
  await expect(page.getByRole('button', { name: `Hi, ${username}` })).toBeVisible();
  const endTime = new Date().getTime();
  log.info(`${workerIndex} login ${username} ${email} - elapsed: ${endTime - startTime}`);
}

B:  - 1.4k driveway.ts  typescript          edit  dos | 28: 0  All
          Type ? for help

-- *Ediff Control Panel*   diff 4 of 4        Quick Help
```

1) login is the first implemention.
2) login_qe is the improved implementation, based on loop test results.

Running the test with Webkit browser on Ubuntu

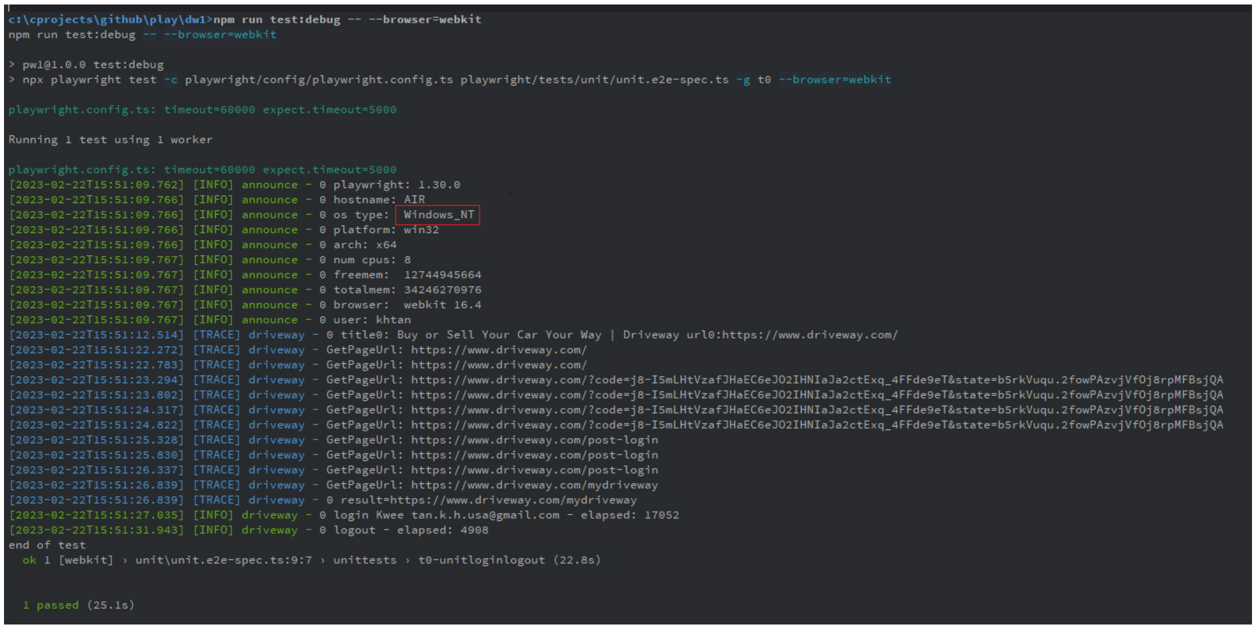
It is interesting that by using a different (slower?) platform the timing issues were more easily seen and dealt with. For example, observe the large number of GetPageUrl retries compared with the Windows platform.



```
khtan@kwee1:~/cprojects/github/play/dw1$ npm run test:debug -- --browser=webkit

> pw1@1.0.0 test:debug /home/khtan/cprojects/github/play/dw1
> npx playwright test -c playwright/config/playwright.config.ts playwright/tests/unit/unit.e2e-spec.ts -g t0 "--browser=webkit"

playwright.config.ts: timeout=900000 expect.timeout=10000

Running 1 test using 1 worker

playwright.config.ts: timeout=900000 expect.timeout=10000
     1 [webkit] › unit/unit.e2e-spec.ts:9:7 › unittests › t0-unitloginlogout
[2023-02-22T11:09:40.136] [INFO] announce - 0 playwright: 1.31.0
[2023-02-22T11:09:40.138] [INFO] announce - 0 hostname: kwee1
[2023-02-22T11:09:40.139] [INFO] announce - 0 os type:  Linux
[2023-02-22T11:09:40.139] [INFO] announce - 0 platform: linux
[2023-02-22T11:09:40.139] [INFO] announce - 0 arch: x64
[2023-02-22T11:09:40.141] [INFO] announce - 0 num cpus: 4
[2023-02-22T11:09:40.141] [INFO] announce - 0 freemem:  5375508480
[2023-02-22T11:09:40.141] [INFO] announce - 0 totalmem: 8263618560
[2023-02-22T11:09:40.142] [INFO] announce - 0 browser:  webkit 16.4
[2023-02-22T11:09:40.143] [INFO] announce - 0 user: khtan
[2023-02-22T11:09:55.144] [TRACE] driveway - 0 title0: Buying New & Used Cars | Driveway url0:https://www.driveway.com/
[2023-02-22T11:10:01.451] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:01.951] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:02.451] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:02.952] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:03.453] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:03.953] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:04.454] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:04.956] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:05.457] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:05.957] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:06.458] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T11:10:06.960] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:07.467] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:07.968] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:08.468] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:08.969] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:09.470] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:09.972] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:10.473] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:10.974] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:11.474] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:11.975] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:12.476] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:12.976] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=CeKc3Dut9OPAlR5BFfS6o9NBNN5oEJTA1EqS4Aok9hM
Z1&state=FgMouOjxN~785rNNj1EgzqfHP7DcoltC
[2023-02-22T11:10:13.477] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:13.981] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:14.482] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:14.982] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:15.484] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:15.986] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:16.502] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:17.002] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:17.503] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:18.004] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:18.512] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:19.012] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:19.514] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:20.014] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:20.514] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:21.015] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T11:10:21.516] [TRACE] driveway - GetPageUrl: https://www.driveway.com/mydriveway
[2023-02-22T11:10:21.516] [TRACE] driveway - result=https://www.driveway.com/mydriveway
[2023-02-22T11:10:21.519] [TRACE] driveway - 0 title2: [object Promise] url2:https://www.driveway.com/mydriveway
[2023-02-22T11:10:22.361] [INFO] driveway - 0 login Kwee tan.k.h.usa@gmail.com - elapsed: 41730
[2023-02-22T11:10:27.630] [INFO] driveway - 0 logout - elapsed: 5268
end of test
  ✓  1 [webkit] › unit/unit.e2e-spec.ts:9:7 › unittests › t0-unitloginlogout (50.6s)
  1 passed (53.6s)
```

Other ways of exposing timing issues include using a cpu killer app to slow one's machine down to find the failure point.

Running the test with Webkit browser on Windows



```
c:\cprojects\github\play\dw1>npm run test:debug -- --browser=webkit
npm run test:debug -- --browser=webkit

> pw1@1.0.0 test:debug
> npx playwright test -c playwright/config/playwright.config.ts playwright/tests/unit/unit.e2e-spec.ts -g t0 --browser=webkit

playwright.config.ts: timeout=60000 expect.timeout=5000

Running 1 test using 1 worker

playwright.config.ts: timeout=60000 expect.timeout=5000
[2023-02-22T15:51:09.762] [INFO] announce - 0 playwright: 1.30.0
[2023-02-22T15:51:09.766] [INFO] announce - 0 hostname: AIR
[2023-02-22T15:51:09.766] [INFO] announce - 0 os type:  Windows_NT
[2023-02-22T15:51:09.766] [INFO] announce - 0 platform: win32
[2023-02-22T15:51:09.766] [INFO] announce - 0 arch: x64
[2023-02-22T15:51:09.767] [INFO] announce - 0 num cpus: 8
[2023-02-22T15:51:09.767] [INFO] announce - 0 freemem:  12744945664
[2023-02-22T15:51:09.767] [INFO] announce - 0 totalmem: 34246270976
[2023-02-22T15:51:09.767] [INFO] announce - 0 browser:  webkit 16.4
[2023-02-22T15:51:09.767] [INFO] announce - 0 user: khtan
[2023-02-22T15:51:12.514] [TRACE] driveway - 0 title0: Buy or Sell Your Car Your Way | Driveway url0:https://www.driveway.com/
[2023-02-22T15:51:22.272] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T15:51:22.783] [TRACE] driveway - GetPageUrl: https://www.driveway.com/
[2023-02-22T15:51:23.294] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=j8-I5mLHtVzafJHaEC6eJO2IHNIaJa2ctExq_4FFde9eT&state=b5rkVuqu.2fowPAzvjVfOj8rpMFBsjQA
[2023-02-22T15:51:23.802] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=j8-I5mLHtVzafJHaEC6eJO2IHNIaJa2ctExq_4FFde9eT&state=b5rkVuqu.2fowPAzvjVfOj8rpMFBsjQA
[2023-02-22T15:51:24.317] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=j8-I5mLHtVzafJHaEC6eJO2IHNIaJa2ctExq_4FFde9eT&state=b5rkVuqu.2fowPAzvjVfOj8rpMFBsjQA
[2023-02-22T15:51:24.822] [TRACE] driveway - GetPageUrl: https://www.driveway.com/?code=j8-I5mLHtVzafJHaEC6eJO2IHNIaJa2ctExq_4FFde9eT&state=b5rkVuqu.2fowPAzvjVfOj8rpMFBsjQA
[2023-02-22T15:51:25.328] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T15:51:25.830] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T15:51:26.337] [TRACE] driveway - GetPageUrl: https://www.driveway.com/post-login
[2023-02-22T15:51:26.839] [TRACE] driveway - GetPageUrl: https://www.driveway.com/mydriveway
[2023-02-22T15:51:26.839] [TRACE] driveway - 0 result=https://www.driveway.com/mydriveway
[2023-02-22T15:51:27.035] [INFO] driveway - 0 login Kwee tan.k.h.usa@gmail.com - elapsed: 17052
[2023-02-22T15:51:31.943] [INFO] driveway - 0 logout - elapsed: 4908
end of test
  ok 1 [webkit] › unit\unit.e2e-spec.ts:9:7 › unittests › t0-unitloginlogout (22.8s)


  1 passed (25.1s)
```

This improved version is currently running an hourly loop and at the point of writing, has been passing for 19 hours.

Table of login times

So in summary, it appears that login varies between 20s and 40s, with Chromium being faster and Webkit being slowest. Since the Lithia team develop on Mac, I would be interested to how the Safari browser behaves.

Time to Login (s)	Chromium	Firefox	Webkit
Win 10	19	24	38
Ubuntu on WSL2	28	42	40

List of good engineering practices

This demo project also shows a number of good engineering practices that help maintain the quality of our tests and the speed to diagnose problems

1. Use third party libraries instead of re-investing the wheel

Eg 1: log4js for logging

Eg 2: retry-ts for retry

Retrying is a pattern that recurs in tests. Before the ts-retry library is used, it is essential to 'try it out'.

tsretry.spec.ts is a simple set of tests to investigate how to use ts-retry and provide a working implementation for others to learn from.

A good way to ensure that the library adopted is of high quality is to code a solution by hand and then compare it with the library.

For example, driveway.ts/login_qe1 login uses a custom retry loop and driveway.ts/login_qe uses ts-retry.

Both functions are tested on a common set of tests to ensure the library performs as well or better

2. Logging and breadcrumbing

This is especially important for Gui and Integration tests because unexpected failures can occur at many points along the way. Breadcrumbing provides a trail to help diagnose the problem. This is in comparison with a unit test that typically will fail very simply.

3. Parallelization

When parallelization is supported right from the beginning, it becomes easier to add parallelization as an engineering solution when the need arises. For example, the logs in the tests all identify the workerIndex. Can you imagine how much work to refactor a non parallel implementation just because we wish to write a set of parallel tests?

4. Rapid diagnosis of functionality

Because each functional action is always accompanied by its own unit test, as soon as the engineer suspects issues in one or a few functions, he can independently run those unit tests to either confirm the failure or move to another potential source of problem. Contrast this with a failing testcase that has to be commented out in areas just to target one failure section.

5. Default style checking

This project deliberately uses eslint and other tools to help engineers concentrate on their code and not on style issues etc. The tools also teach and advise engineers about problematic coding patterns.

This is a decentralized way of working that works because no resource is needed to maintain a set of standards, or to enforce them.

6. Support for teamwork

By supporting many orthogonal issues (logging, breadcrumbing, parallelization, test suites, default style checking etc) up front, it is easier to enable a new member of the team to join and become productive. Similarly, when tests get sub-contracted to off-site teams, they have a working model to follow.