

Automation test summary

Intro

This document summarises the approach taken to build an example of an automation test suite for a webapp page. This discuss several element of the approach on a medium level of detail, however for actual projects a test strategy document will be used with high level of detail.

First step in the process is test planning which targets the goal of getting familiar with the tool and do the required analysis to define the system, the scope of test, what's out of scope, what are the features this system provide and if it will need special requirements from the automation tool. Like for example in case we would have images comparisons, maps widgets, video calling involved etc..

The main aspects of the system and testing environment is defined as below as part of exploratory testing aiming to:

- check the validity of the system functionality.
- the stability of the system.
- The usability of different work flows.
- If any bugs exists (which led to the early discovery of a bug in the system check the reporting section).

System:

The awesome Q/A tool

Description:

A Questions/Answers form provides several features of Q/A entry, sorting and submission.

Testing medium:Desktop Web browsers (Chromium & Safari)

Out of scope: Mobile browsers

Features:

- F1.Viewing existing questions
- F2.Clicking on questions to see the answers
- F3.Adding new questions
- F4.Sorting questions Alphabetically
- F5.Removing questions
- F6.Displaying number of questions
- F7.Showing a label for empty questions list
- F8.Hiding sorting & Remove questions buttons when no questions in the list
- F9.Display of a tooltip on hover on titles with descriptions for each section
- F10.Validations for empty question/answer text fields

Adjustable answers text field (Not Automated)

After the exploratory testing of the Web page and defining its feature now comes the time to go for *Test case design phase* where I focused on the following:

- Covering the most features with the convenient no. of test case.
- Evaluating the value of automating some features for example, the last feature “Adjustable answers text field” is a low value feature which is hard to automate so it was dis-regarded from the automation test suite.
- Organising test cases groups
 - For groups practically we need at least two types of groups one is the MVP for happyflows which will make sure that the vital features are working and this is very useful if it's part of regression or smoke tests when we need to quickly check that this part of system is still functioning,
 - The other group will aim for a higher coverage for edge cases and negative scenarios as well as long as they are valuable enough and this is how the below groups are created

Test cases:

MVP:

- *AddNewQuestion*: Add a Q/A and validate it in the list, validate the total no. of questions updated.
- *SortQuestionsList*: Sort Q/As in the list and validate the order.
- *RemoveQuestionsList*: Remove Q/As from the list, validate no questions label and validate the total no. of questions of questions updated, sorting/removing buttons are hidden.

Extensive list:

- *AddNewQuestionOnEmptyList*:
Empty the Q/As list then add a Q/A, validate the Q/A, sorting & removing are enabled again, validate total no. of questions updated.
- *CheckEmptyTextFieldsValidations*:
Enter a Q/A with an empty Question text field, validate an error is triggered and user is blocked
- *CheckEmptyTextFieldsValidations*:
Enter a Q/A with an empty Answers text field, validate an error is triggered and user is blocked
- *fillTxtFieldsWithEmptySpaces*:
Enter a space in question & answer text fields
- *ValdiateHoverTooltips*
Hover over the titles and validate the prompts

- ValdiatePageContents: Validate the page contents ex. Title, headers, etc.

Coverage:

After designing test cases a vital step to have a holistic view over the created design to see in terms of coverage two things. First the percentage of the coverage of testcases vs. features. Second most importantly to check the distribution of testcases over the scope which helps a lot to distribute testability of a critical feature over different test cases.

Requirement traceability	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
<i>AddNewQuestion</i>	x	x	x							
<i>SortQuestionsList</i>	x		x	x						
<i>RemoveQuestions List</i>	x				x	x	x	x		
<i>AddNewQuestionOnEmptyList</i>	x		x		x	x		x		
<i>CheckEmptyTextFieldsValidations</i>	x					x				x
<i>fillTxtFieldsWithEmptySpaces</i>	x									x
<i>ValdiateHoverTools</i>	x								x	
ValdiatePageContents	-	-	-	-	-	-	-	-	-	-

Implementation:

As a conclusion of the planning and designing phases no special requirements are needed from the automation tool to be used which provides flexibility of choice. Playwright is one of the convenient trendy framework that is easy to use and the same time has a big community of support online and widely used plus being an open source comes as a good option for such a task. After double checking that the needed requirements match the tool capabilities. Playwright was chosen.

For detailed information about how to setup and run tests please have a look in the project readme file. (qaReadMe.md)

Project structure

Project structure is built taking in consideration the simplicity and the future maintainability of scripts so I have used multiple elements to centralise objects, variables, functions in centralised files.

The main components of the project are

Tests

all test-cases are divided into two TS scripts follows the same grouping mentioned before -for convenience-

POM

A page object model for the only page we have in scope “QAform” centralising the page objects, their initialization and any repetitive function used by different scripts.

Fixtures

A fixture “fixtureQAformPage” which extends the original Playwrite.test with more objects and in this case I used to include the POM “QAform” globally in different tests by importing the extended fixture version of .test which will automatically include the POM items

TestData

Serving the goal of centralising variables all the test data used by test cases are added to a json file “testdataV0.1” which represents one version of test data which we can later have multiple ones for multiple environments.

Config file

The default playwright.config.ts file is used to control the global configurations of tests like which browsers are running the execution, no. of parallel workers, no. of retries on failure and so on

Test-results

This folder is dedicated to the test results and in this projects it will contain also screenshots from failed execution runs

Execution:

A python http server is run locally with

As the tests are relatively simple and fast to execute, the execution strategy isn't making a big impact in this case but ideally when working on a real projects usually execution will face challenges related to flakiness or dependencies between scripts which will encourage the sequential execution. In other cases tests can be long or taking much time due to waiting server actions which will encourage the parallel execution. So for this project I have implemented both for the sake of demonstration.

Execution is designed to as follows:

- MVP-Happyflows→ sequential
- ExtensiveTest → Parallel







Retries are set to 3 times in the config file which will assure that a failing test case is a result of 3 consecutive failures, also using two different browsers helps it increases the coverage as well to scope the issue with a failing test case if it's browser related or test related.

For parallel the execution the number of parallel instances(workers) is taken up to 6 instances considering having 5 testcases in parallel and 3 that run sequentially. on the mentioned browsers

Reproting:

Reporting relies on main 4 pillars. Playwright default reports, screenshots on failure, videos on failure and tracing

Clear html reports are used with categories for pass, failure, flaky, skipped test cases. Divided into the two logical groups mentioned above.

<input type="text"/>		All 16	Passed 13	✖ Failed 2	⚠ Flaky 1	Skipped 0
ExtensiveTest.spec.ts						58.4s
✖	ExtensiveTestSuite › fillTxtFieldsWithEmptySpaces	chromium				12.0s
ExtensiveTest.spec.ts:68  						
✖	ExtensiveTestSuite › fillTxtFieldsWithEmptySpaces	webkit				7.2s
ExtensiveTest.spec.ts:68  						
⚠	ExtensiveTestSuite › AddNewQuestionOnEmptyList	webkit				11.0s
ExtensiveTest.spec.ts:29  						
✓	ExtensiveTestSuite › AddNewQuestionOnEmptyList	chromium				4.5s
ExtensiveTest.spec.ts:29						
✓	ExtensiveTestSuite › CheckEmptyTextFieldsValidations	chromium				4.7s
ExtensiveTest.spec.ts:48						
✓	ExtensiveTestSuite › CheckEmptyTextFieldsValidations	webkit				3.7s
ExtensiveTest.spec.ts:48						
✓	ExtensiveTestSuite › ValdiatePageContents	chromium				4.4s
ExtensiveTest.spec.ts:85						
✓	ExtensiveTestSuite › ValdiatePageContents	webkit				4.0s
ExtensiveTest.spec.ts:85						
✓	ExtensiveTestSuite › ValdiateHoverToolTips	chromium				4.5s
ExtensiveTest.spec.ts:103						
✓	ExtensiveTestSuite › ValdiateHoverToolTips	webkit				2.3s
ExtensiveTest.spec.ts:103						
MVP-happyflows.spec.ts						12.1s
✓	MVP-HappyFlows › AddNewQuestion	chromium				4.5s
MVP-happyflows.spec.ts:17						
✓	MVP-HappyFlows › AddNewQuestion	webkit				1.2s
MVP-happyflows.spec.ts:17						
✓	MVP-HappyFlows › SortQuestionsList	chromium				1.2s
MVP-happyflows.spec.ts:41						
✓	MVP-HappyFlows › SortQuestionsList	webkit				1.8s
MVP-happyflows.spec.ts:41						
✓	MVP-HappyFlows › RemoveQuestionsList	chromium				2.5s
MVP-happyflows.spec.ts:60						
✓	MVP-HappyFlows › RemoveQuestionsList	webkit				899ms
MVP-happyflows.spec.ts:60						

On failure screenshots are taken and moreover videos are available to show the exact flow on failure. Also with the 3 retries configured before. Data is collected in each run

and can be viewed in the report as part of test case run detail page.

chromium

✖ Run ✖ Retry #1 ✖ Retry #2 ✖ Retry #3

Errors

Error: expect(received).toBe(expected) // Object.is equality
Expected: 1
Received: 2

78 | //validate Q/A isn't accepted by the system
79 | const questionsList = await page.\$\$('.question__question');
> 80 | expect (questionsList.length).toBe(1);
 | ^
81 | expect (qaForm.sidebarText).toContainText('2');
82 |
83 | });

at /Users/momenelsawy/Downloads/qa-engineer-assignment-master/tests/ExtensiveTest.spec.ts:80:35

Test Steps

> ✓ Before Hooks	1.5s
> ✓ page.goto(http://127.0.0.1:5500/) — ExtensiveTest.spec.ts:71	364ms
> ✓ locator.fill(id=question) — POM/QAform.ts:38	88ms
> ✓ locator.fill(id=answer) — POM/QAform.ts:41	19ms
> ✓ locator.click(btn-success) — POM/QAform.ts:44	149ms
> ✓ page.\$\$('.question__question') — ExtensiveTest.spec.ts:79	13ms
> ✖ expect.toBe — ExtensiveTest.spec.ts:80	3ms
> ✓ After Hooks	884ms

Screenshots

Traces are turned on in the execution phase to have extra layer of details action by action about the flow of a test run. Which later can be viewed by a trace viewer. All the screenshots and videos from different runs can be found in Test results folder.

Conclusion:

After several executions the results concluded the following:

- The system has one bug mentioned in the next section which was discovered in the exploratory testing and an automated test cases is created for it for future validation
- Minor instability discovered in the system affecting some test cases. By looking into the data there is no clear pattern for it however it can be investigated by further executions.
- Test cases are built with the simplest form relatively for the simplicity of the system however more dynamic implementations can be used for real life scenarios. For example. Sorting scenario can be filled in with variable inputs and validated by a dynamic function checks for the alphabetical sorting of the list regardless the input.

Issues found:

As a user It's expected to not be able to create an empty question or answer in Q/A tool in order to prevent a faulty submissions of incomplete Q/A

Description: Both input fields for Questions & Answers have validations to not allow empty submissions of those fields however users can bypass this validation by typing in a space on the keyboard.

Reproduction:

1. Navigate to the Q/A main page
2. In the Question text field type a space on keyboard
3. In the Answer text field type a space on keyboard
4. Click on "Create question" button

Expected: An error message should be displayed mentioning that those fields can't be empty and user shouldn't be able of creating this Q/A

Actual: Q/A is created successfully and added to the list with empty Q&A fields