Loop Technical Evaluation

Author: Ken Riley kenr@nodots.com

Date: 2024.12.05

Introduction

Create a Playwright-driven test suite that leverages data-driven techniques to minimize code duplication and improve scalability. By driving test scenarios from a JSON object, we can dynamically adapt each test case without repeating code, ensuring a clean and maintainable structure as new cases are added.

Implementation Details

TypeScript types and interfaces

There is a hierarchy of AsanaBoard -> AsanaSwimlane -> AsanaStory and corresponding types that add testing-specific data.

```
export interface AsanaStory {
   title: string
   description: string
   tags: string[]
}

export interface AsanaSwimlane {
   order: number
   kind: ASANA_SWIMLANE_KIND
   stories: AsanaStory[]
}

export type AsanaStoryTestData = AsanaStory & { included: boolean }
   export type AsanaSwimlaneTestData = Omit<AsanaSwimlane, 'stories'> & {
     stories: AsanaStoryTestData[]
}
```

Data-Driven Testing

All test data comes from test-data.ts which implements the tests described in the requirements document. This data in turn drives which tests are performed using the type system described above.

DOM Parsing: getSwimlanes function

There is a single DOM-parsing function that retrieves the Swimlanes from the given page. This is heavy on specific selectors ATM, but, we need to infer relationships between tasks and swimlanes to ensure:

• Each test is preformed on the correct swimlane

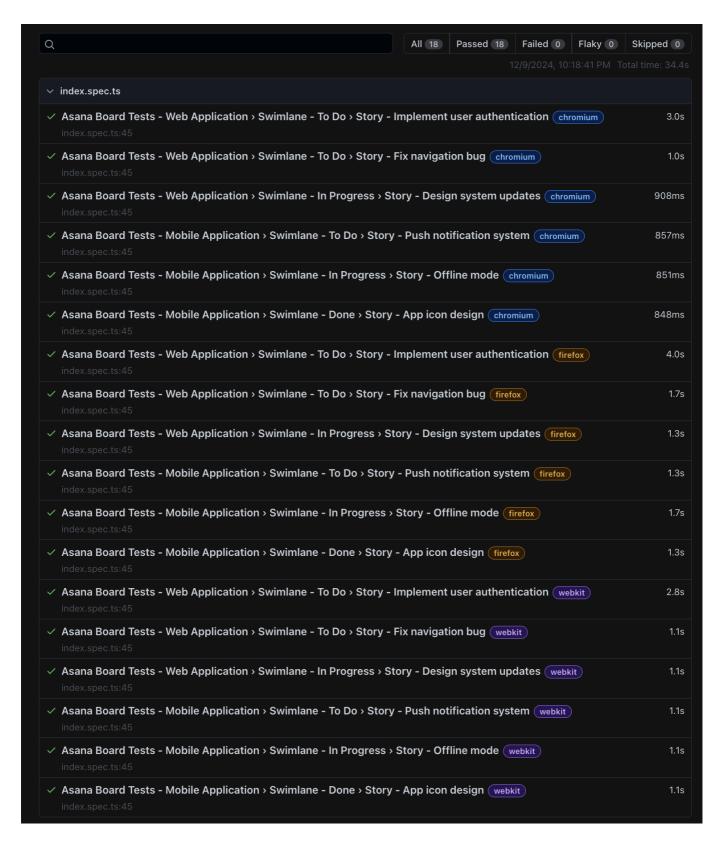
• That tests for the existence of things are carried out only on relevant portions of the DOM. E.g., we need to check for both for "Design" as a tag and "Design system updates" as a task title.

Challenges and Solutions

The key challenge in testing this code is isolating discreet sections of the DOM to ensure we are checking for elements/strings in the right place.

While it is a best-practice to test only user-visible behavior, achieving this required some DOM parsing. To isolate this as much as possible I wrote a <code>getSwimlanes</code> helper function that uses very basic selectors that is run before each test. If there is a change in the DOM of the target page, this will at least break cleanly and in a way that is easy to fix.

Results



Recommendations

Retrospectively, I could have done a better job separating the JSON test data from the implementation. There should have been a separate JSON file which would have both made it easier to maintain the test suite and to potentially generate new test suites for JSON files generated by a tool.

I would consider adding the Asana TypeScript types to improve the getSwimlanes function because doing so *might* make it easier to parse the Asana data and it is reasonable to assume I think that all Asana instances will implement the official Asana types.

The downside is that it creates a third-party dependency, which is why I chose to create very simple types that describe the how Boards/Swimlanes/Stories/Tags are related in the presentation layer.