SauceDemo & Petstore Report

Automation tool: Playwright(JS)

To run the tests use **npx run e2e** or **npm run api.**

Playwright was chosen since it was the preferred automation tool in this assignment. It is also a framework I'm currently learning so I thought this was a perfect moment to do some work and learn at the same time.

UI testing

The UI testing is done following the page object model. This ensures that locators and page specific functions are easily accessible in one file. POM - model makes the test more maintainable.

The locators used in the UI testing are almost exclusively data-test attributes. I prefer to use test specific locators if available.

All tests are created to pass to reflect the current state of the demo site. Generally I don't commit tests that I know will fail. That's the reason I did not run the test on each of the users supplied on the Login Page.

From the scenarios outlined in the test plan in some cases I had to break the automated tests into smaller tests. I want each test to only test one specific thing to keep the tests small and maintainable.

Test data was in this case supplied on the login page and was constantly the same in the store. So there was no real need to worry about test data management.

The tests are running on both Desktop (Safari,Chrome,Firefox) & Mobile(Safari,Chrome).
For the test on the shop/inventory page I included an authentication setup that set the cookie that let you access the store before every test.

For this whole suite of tests I can run the tests in parallel.

API Testing

The API testing was made by making scenarios based on the documentation on the swagger site. This might have been a mistake since the API's documentation is highly inconsistent with how it actually functions.

POST/PUT requests can send in parameters with the wrong type/ value. It accepts JSON objects without required values. In all cases I tested it saves the data anyway. This has caused me to write tests that I know fail.

For the API tests I have created a helper file to handle data before and after every test. The helper file also contains functions to help keep the tests shorter in the spec files. Similar to the page object model used for the E2E tests.

Thanks to the data management I was able to run all API tests in parallel mode.