Automation Testing for an Ad Platform

Overview:

The client for this project works in social media advertisement. To their dismay they were noticing that they were not able to offer their customers an optimal experience. They were plagued by an increase in the number of bugs that were showing up in their application, and without a dedicated testing team addressing those, matters were fast spinning out of control. They wanted Mindfire to assist them in overcoming this hurdle by creating a robust testing framework that would cover all the different types of testing they were in need of i.e. Functional UI testing, Network/API testing and Visual testing as their application was graphic- intensive.

Client details

Client Name: Confidential **|** **Industry:** Media & Entertainment **|** **Location**: USA

Technologies

Cypress

Project Description:

The client wanted Mindfire to offer them a robust testing framework covering a range of solutions that could help them overcome the glitches that their application was throwing up. Specifically, they wanted

1) the framework to be based on JavaScript as their own development framework was based on that. They wanted this to make life easier for the developers and testers in the future.

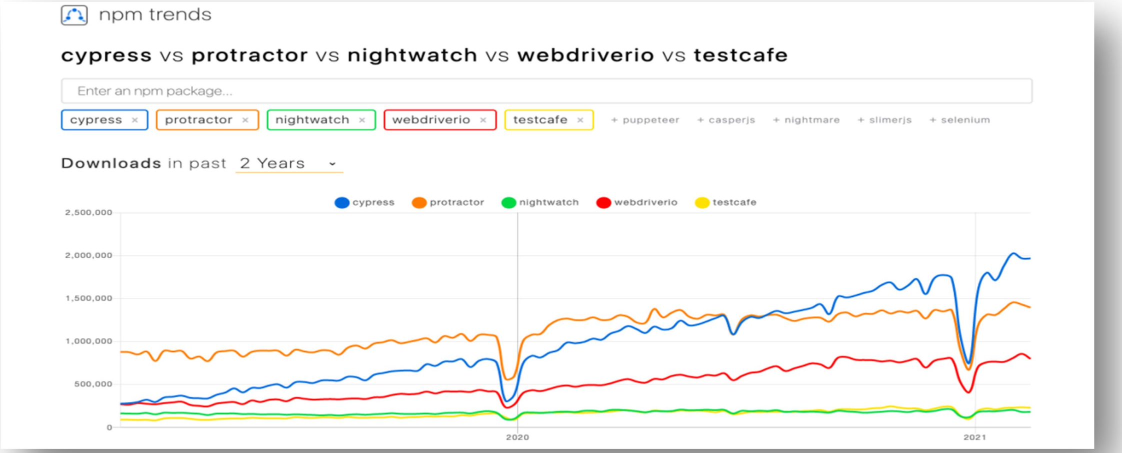
2) one framework for all different type of testing (Functional, Non-functional, API, Visual)

3) support across multiple browsers along with support for BDD cucumber so that even a user from a non-technical background would be able to understand what the test cases were about.

R&D for the Best Solution:

After we spoke to the client and had a look at the website, we began our search for a solution which would fulfill all the needs of the client. There were 2 conditions laid before us. The first one was the meet the factors mentioned above and the second one was to keep things as simple as possible, abiding by their guiding philosophy of KISS - Keeping things Simple & Stupid.

We evaluated multiple tools, had meetings to discuss their pros and cons of each and ended up shortlisted a few after the vetting process. Finally, Cypress emerged as the best choice that seemed to perfectly fit the client's needs. is a JavaScript based end-to-end testing framework, it can test anything that runs on a web browser.



cypress vs protractor vs nightwatch vs webdriverio vs testcafe

Downloads in past 2 Years

There were some compelling reasons for our choice of Cypress. In the world of Automation, Cypress is comparatively new, and it removes certain restrictions which we have observed, in general, in other automation tools.

We felt Cypress was the perfect choice because

1) The first time setup process is very simple: no complex installations required and no need to maintain chrome-driver or any other browser driver.

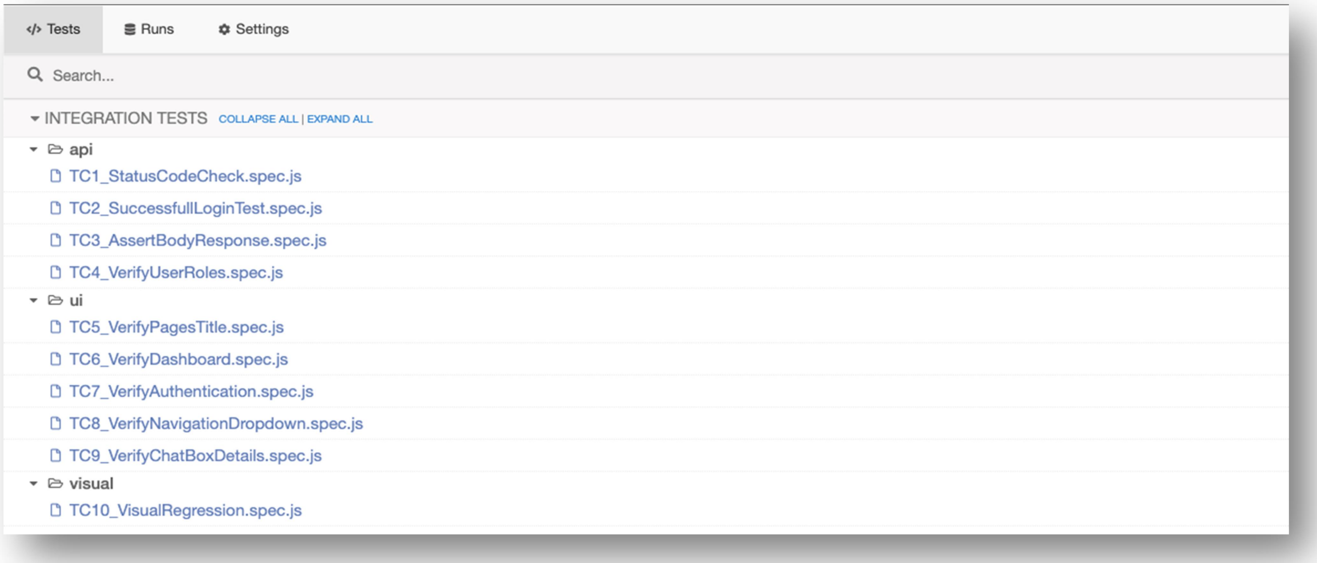
2) Since Cypress directly runs on a browser, we do not need to add any “wait” as it is handled automatically and there is an almost zero chance of flaky tests

3) Videos are automatically generated for each test spec file, which is a great way to share the application behaviour with other members of your team.

4) Support to integrate applications like Jenkins, Slack, GitHub.

5) It is a JavaScript based tool and supports BDD too.

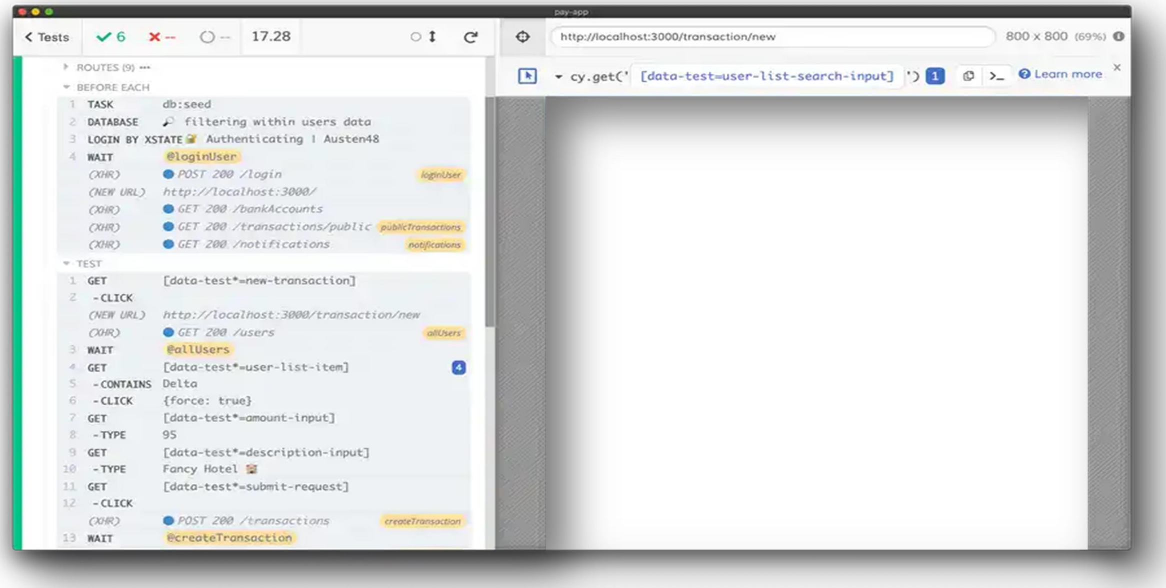
6) Once Cypress is installed you get a pre-defined framework which is actually useful and you can instantly get started with your project.



```
INTEGRATION TESTS
api
  TC1_StatusCodeCheck.spec.js
  TC2_SuccessfullLoginTest.spec.js
  TC3_AssertBodyResponse.spec.js
  TC4_VerifyUserRoles.spec.js
ui
  TC5_VerifyPagesTitle.spec.js
  TC6_VerifyDashboard.spec.js
  TC7_VerifyAuthentication.spec.js
  TC8_VerifyNavigationDropdown.spec.js
  TC9_VerifyChatBoxDetails.spec.js
visual
  TC10_VisualRegression.spec.js
```

Test cases listed in Cypress GUI

After you write your first real test spec, you can run it using the Cypress command or Cypress GUI. We prefer running via Cypress GUI because if you are running the test for the first time and in case it fails, you can come back to your code, make necessary updates and Cypress GUI automatically realizes this update and re-starts the test. So during complex test scenarios, this saves a lot of time when the user wants to try multiple things.



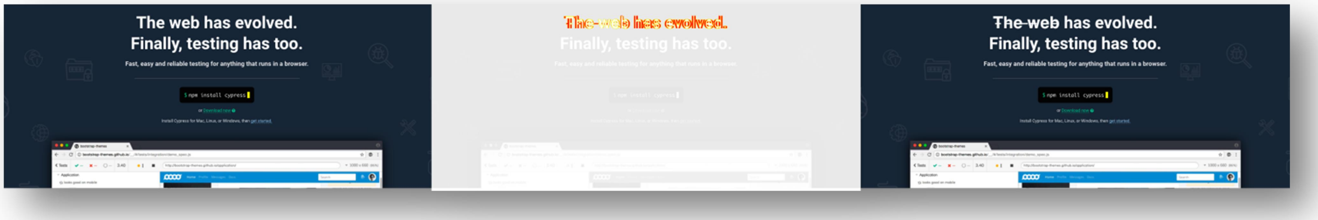
```
http://localhost:3000/transaction/new

cy.get(' [data-test=user-list-search-input] ') 1
```

```
ROUTES (9) •••
BEFORE EACH
1  TASK              db:seed
2  DATABASE          filtering within users data
3  LOGIN BY XSTATE   Authenticating | Austen48
4  WAIT              @loginUser
   (XHR)             POST 200 /login                        loginUser
   (NEW URL)         http://localhost:3000/
   (XHR)             GET 200 /bankAccounts
   (XHR)             GET 200 /transactions/public   publicTransactions
   (XHR)             GET 200 /notifications             notifications
TEST
1  GET               [data-test*=new-transaction]
2   -CLICK
   (NEW URL)         http://localhost:3000/transaction/new
   (XHR)             GET 200 /users                          allUsers
3  WAIT              @allUsers
4  GET               [data-test*=user-list-item]                    4
5   -CONTAINS        Delta
6   -CLICK           {force: true}
7  GET               [data-test*=amount-input]
8   -TYPE            95
9  GET               [data-test*=description-input]
10  -TYPE            Fancy Hotel
11 GET               [data-test*=submit-request]
12  -CLICK
   (XHR)             POST 200 /transactions        createTransaction
13 WAIT              @createTransaction
```

Snapshot during Cypress test execution

When we created about 10 test cases in the area of functional testing, network based testing and visual regression testing, we ran the entire test suite and overall noticed during the entire process that the test ran faster compared to other automation tools.


The web has evolved.
Finally, testing has too.
Fast, easy and reliable testing for anything that runs in a browser.
$ npm install cypress
The web has evolved.
Finally, testing has too.
Fast, easy and reliable testing for anything that runs in a browser.
$ npm install cypress
The web has evolved.
Finally, testing has too.
Fast, easy and reliable testing for anything that runs in a browser.
$ npm install cypress

Visual Regression where base image and test image are compared pixel by pixel

Learnings:

Cypress is a very good option if you do not mind coding in JavaScript for your project. With Cypress you get that speed and tests run very fast. It all depends on how fast you can complete writing your tests. We did not notice any flaky tests while using Cypress. It was either a pass or a fail. So you end up confident that your tests are robust and reliable.

There is also a feature in Cypress called “Cypress Dashboards” that gives you a dashboard where you can see the reports of your test suits. It also maintains daily history so that the user can gauge the performance as required. Data-security wise, you do not need to depend on any third party for reporting, as this feature is already integrated. Overall Cypress is

- Easy to Setup
- Runs or re-runs very fast
- Low maintenance
- Improves productivity

It seamlessly integrates with the popular apps (e.g. Slack, GitHub) and supports all major browsers