# Senior Test Engineer: Take-home Technical Task Report

### Branko Stojakovic

## Task Repository location

[YTests/cypress/e2e at main · elector101/YTests (github.com)](https://github.com/elector101/YTests/tree/main/cypress/e2e)

## Test Plan SauceDemo

### Introduction

In this section the focus will be on testing approach applied to testing a mock e-commerce website located at <https://www.saucedemo.com/>

### Scope

The resulting test suite will be focused on functional tests regarding the login and product purchasing.

### List of features to be tested

Focus should be on testing:

* Customers log in
* The main shop page
* The cart page
* Adding a product to their cart
* Checkout page
* Checkout finish

## Test Plan API

### Introduction

The focus of this section is to describe approaches and methodologies that will apply to the testing of the Open Weather API located at <https://rapidapi.com/worldapi/api/open-weather13>.

### Scope

Tests will be focused on basic Login and Functional test that will be automated. Main area of concern are:

* Functional testing

Performance testing and Security testing are out of scope for this suite.

### List of tests

The API under test holds 3 methods but we will focus on one. The approach would be replicated across other API methods.

We will be testing:

* Result of a successful call to service method
* Sending in invalid parameters
* Sending in invalid API key values

With these tests we can confirm that the service returns correct values and correct response codes.

## Reflection

### General approach

First step was to analyse the website to be tested and to chose and review the API to be tested with running exploratory testing on both as a basis for code implementation. Website is a simple version of an e-commerce website and the API chosen was the 3 method weather report service.

### Challenges

Working with Cypress for the first time was the main challenge. Not been completely certain about correct configuration, I have placed all test files in the e2e folder. In a proper scenario I would not mix Website and API tests in one project.

### Testability

Both resources were testable with some minor problems. Website had a loading issue that was solved and explained in the comments. API has a limit on the number of calls per month for a free version.

### Test engineering on the systems

For the website, the scope and types of test should be expanded to cover:

* Functional testing – separate set of tests for every specific feature
* Usability testing, Visual testing – discover changes in the UI and how they affect the usability
* Performance testing – test for responsiveness, load baring, stability…
* Security testing – pen testing, simulating attacks, data exposure
* Device compatibility testing – it is important today that sites are responsive and adaptable to different devices
* User Acceptance – main functionalities important for the user
* End to End Testing – full system functionality checks

For the API, tests should be expanded to:

* Full functional testing with all data validated as a result of a method call
* Performance testing – load testing as the priority
* Security testing – essential especially for service open to 3rd parties

### Quality of the test suite

This is evidently a mock test suite that should just give an indication to the examiner about the knowledge and potential skill set of the candidate.

Individual tests should be more detailed and expanded in checking the returning values making sure that all the important return values are present and correct as described in Test engineering section.

### Other comments

* SourceDemo website was giving an error on multiple login test calls returning:  
  *Your page did not fire its load event within 60000ms.*I didn’t delve deeper into the issue, I found a simple solution on <https://github.com/cypress-io/cypress/issues/21213> with inputting *chromeWebSecurity: false* in cypress.config.js
* According to Cypress documentation, slight advantage in running time can be achieved when using Cypress App Actions.
* In the API tests, the URL, API key and other static values should be extracted from hard coded test code into an environment configuration file.