Testing WheelMap.org using Selenium

Matt Walter, Janneke Morin, Justin Garrison

Introduction

Wheelmap is a crowdsourced online map to search, find, and mark wheelchair-accessible places.

The developers use a third-party testing suite called *BrowserStack*, so we didn't have access to their test cases. Thus, we opted to create our own using Selenium, a browser automation package for Python.

Materials and Methods

Basic test running functionality is provided by the standard python library, **unittest.** In order to test the web browser we utilized **Selenium,** a powerful web-based automation tool useful for testing. To display the test results we used **HTMLTestRunner,** an extension to the unittest library.

Results

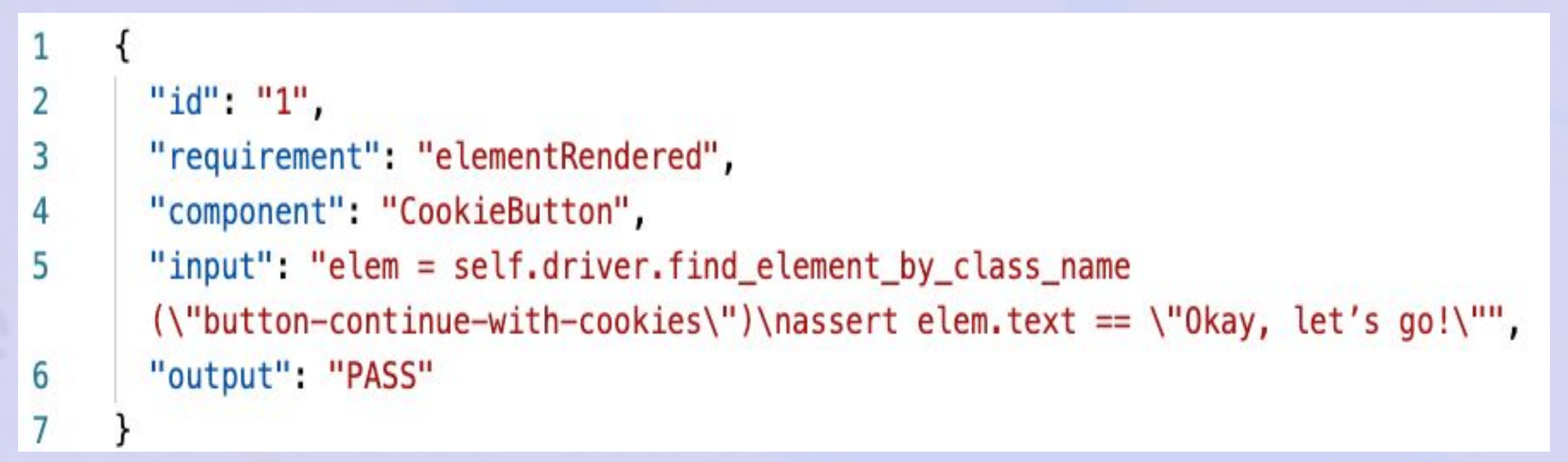
Since Wheelmap is primarily a front-end project, it's important to test the UI of the web-app.

Test cases range from ensuring elements are rendered on screen, to ensuring the elements respond properly to user interactions, *see figure 1.*

Our modified HTMLTestRunner output displays verbose and intuitive test results, *see figure 2.*

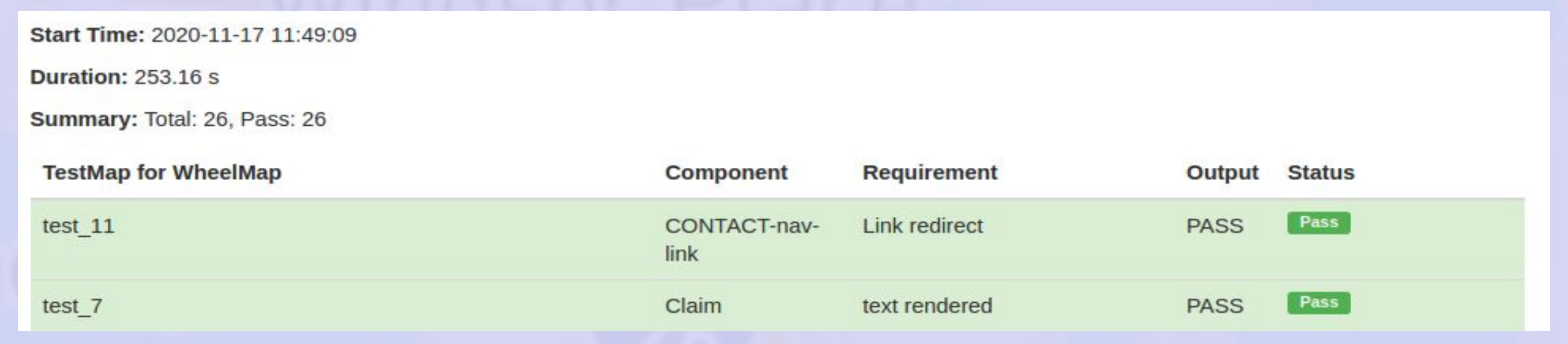
When a test fails HTMLTestRunner distinguishes between errors within the test case and actual test failures, *see figure 3.*

The test framework's process is outlined below.



```
{
  "id": "1",
  "requirement": "elementRendered",
  "component": "CookieButton",
  "input": "elem = self.driver.find_element_by_class_name
  (\"button-continue-with-cookies\")\nassert elem.text == \"Okay, let's go!\"",
  "output": "PASS"
}
```

Figure 1. Example TestCase json file for the initial accept cookies button



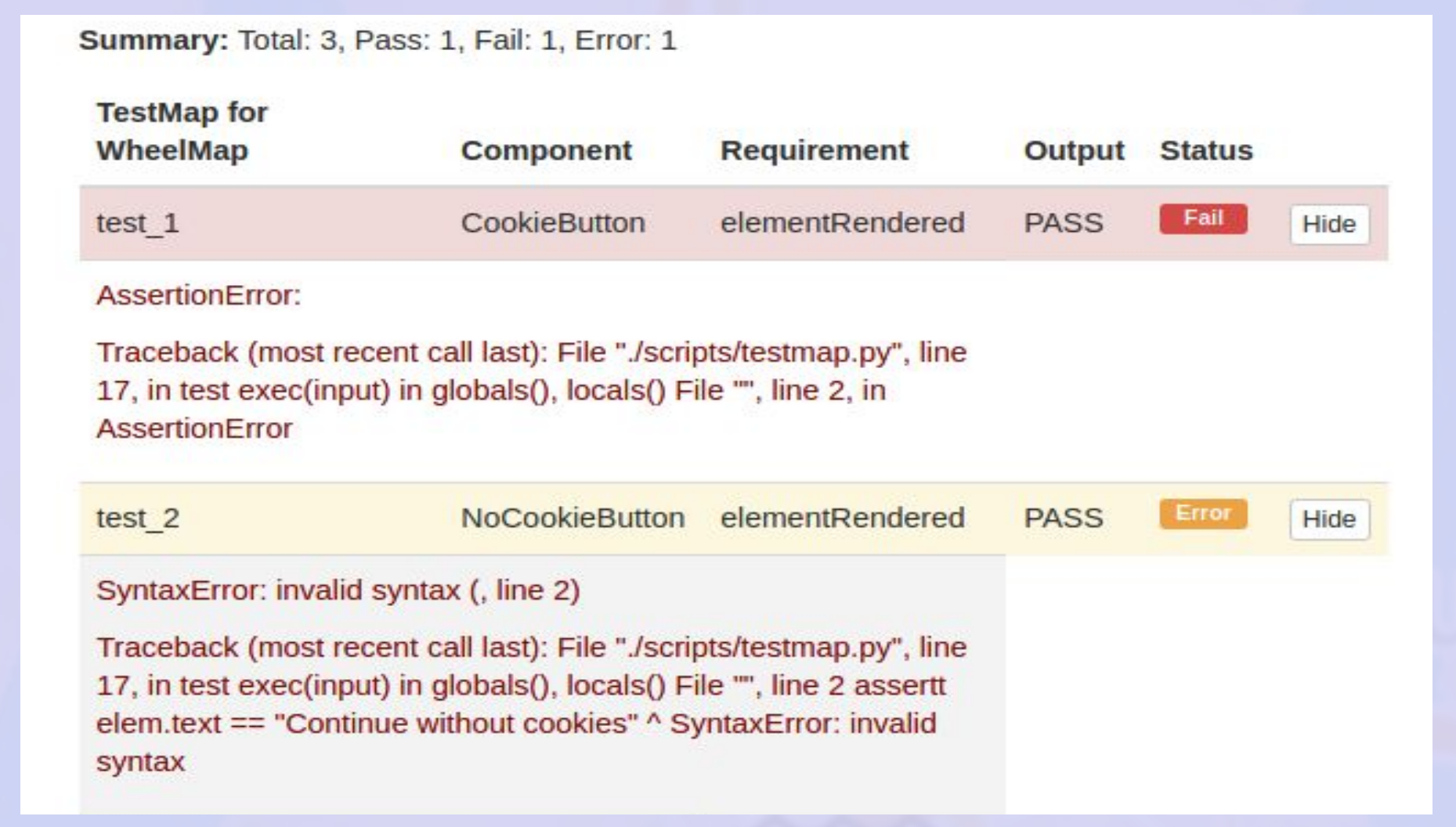
Start Time: 2020-11-17 11:49:09

Duration: 253.16 s

Summary: Total: 26, Pass: 26

TestMap for WheelMap	Component	Requirement	Output	Status
test_11	CONTACT-nav-link	Link redirect	PASS	Pass
test_7	Claim	text rendered	PASS	Pass

Figure 2. HTMLTestRunner output for successful tests



Summary: Total: 3, Pass: 1, Fail: 1, Error: 1

TestMap for WheelMap	Component	Requirement	Output	Status	
test_1	CookieButton	elementRendered	PASS	Fail	Hide

```
AssertionError:

Traceback (most recent call last): File "./scripts/testmap.py", line 17, in test exec(input) in globals(), locals() File "", line 2, in AssertionError
```

TestMap for WheelMap	Component	Requirement	Output	Status	
test_2	NoCookieButton	elementRendered	PASS	Error	Hide

```
SyntaxError: invalid syntax (, line 2)

Traceback (most recent call last): File "./scripts/testmap.py", line 17, in test exec(input) in globals(), locals() File "", line 2 assertt elem.text == "Continue without cookies" ^ SyntaxError: invalid syntax
```

Figure 3. Example of failing test cases.

Conclusions

It may appear trivial to test a web-app's UI, but an "unfriendly" user interface will at best deter users and at worst break the application, so this project proved invaluable to understanding how front-end testing can be utilized in order to ensure a visually-appealing, stable and intuitive design for the user.

While our testing framework works as intended, there are aspects of the framework design that can (should) be improved in the future, such as scalability. The framework will not scale well to a larger number of test cases, this can be mitigated by using a pipeline from test case to the driver.

testCaseX.json

Test case data is stored in a json file within the testCases directory of the project. Test cases have an ID, component, requirement, input (python selenium code), and expected output.

Parser.py

All json test case files are parsed and its data is stored as a testCase object. The parse function returns a list of testCase objects to the driver.

testmap.py

testmap.py is the main driver of the testing framework. Here the list of testCase objects is iterated over and its python test code is executed. The results of the tests are passed to the HTMLTestRunner.

HTMLTestRunner

opens the browser and displays all the test results. It provides the test case ID, component, requirement, the oracle, and the actual output in the form of PASS, FAIL, or ERROR.