Team 7-11

Justin Garrison - Matt Walter - Janneke Morin

Choosing a project

1. wheelmap

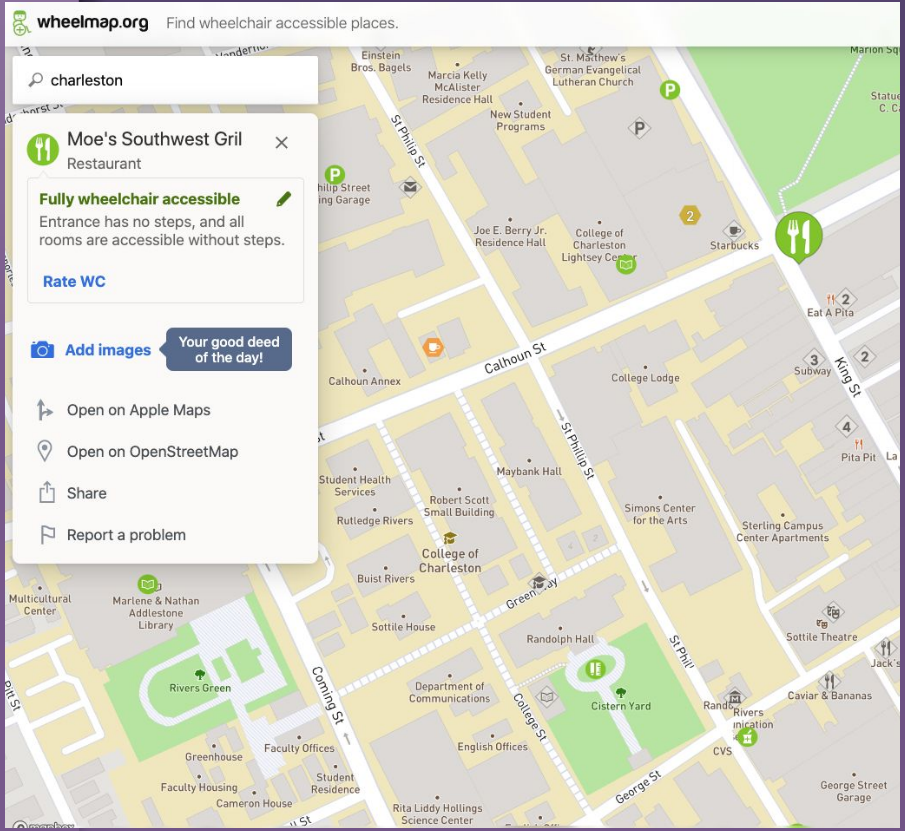
    - Solid documentation
    - Primarily JavaScript
    - Front-end application
    - Great cause

2. CADASTA

3. martus

wheelmap

“Wheelmap.org is an online map to search, find, and mark wheelchair-accessible places.”



Moe's Southwest Gril
Restaurant

Fully wheelchair accessible
Entrance has no steps, and all rooms are accessible without steps.

Rate WC

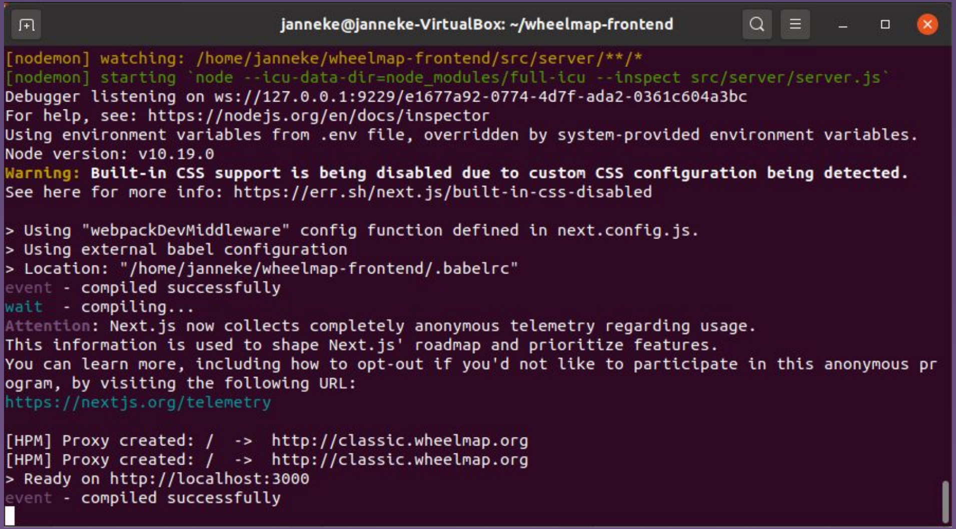
Add images
Your good deed of the day!

Open on Apple Maps
Open on OpenStreetMap
Share
Report a problem

Building Wheelmap

- Relatively simple issues
  - node-package manager

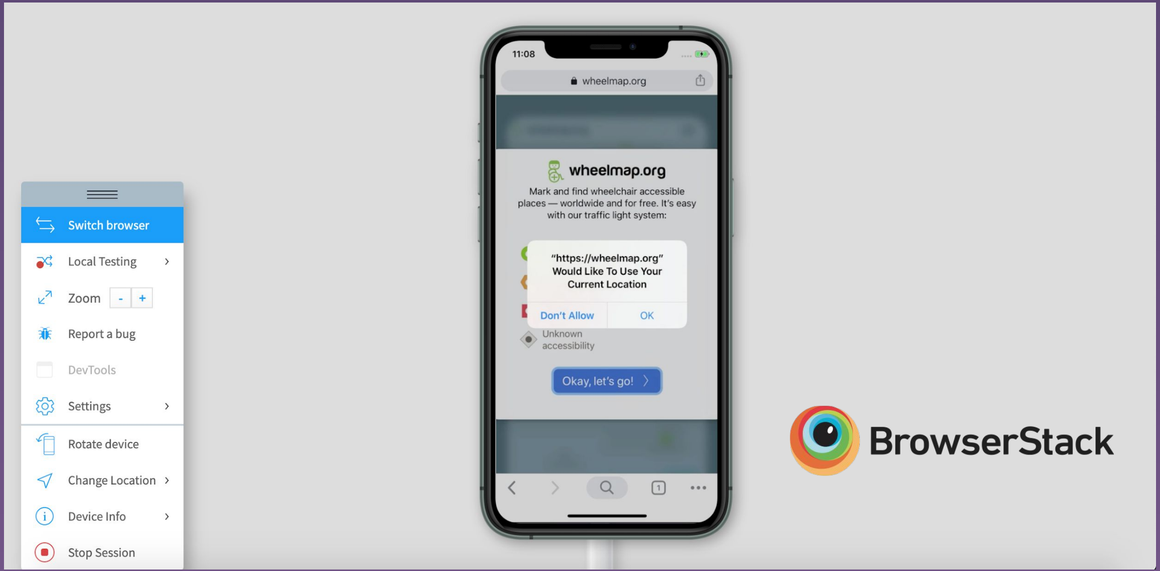
Terminal upon successful compilation


janneke@janneke-VirtualBox: ~/wheelmap-frontend

```
[nodemon] watching: /home/janneke/wheelmap-frontend/src/server/**/*
[nodemon] starting `node --icu-data-dir=node_modules/full-icu --inspect src/server/server.js`
Debugger listening on ws://127.0.0.1:9229/e1677a92-0774-4d7f-ada2-0361c604a3bc
For help, see: https://nodejs.org/en/docs/inspector
Using environment variables from .env file, overridden by system-provided environment variables.
Node version: v10.19.0
Warning: Built-in CSS support is being disabled due to custom CSS configuration being detected.
See here for more info: https://err.sh/next.js/built-in-css-disabled

> Using "webpackDevMiddleware" config function defined in next.config.js.
> Using external babel configuration
> Location: "/home/janneke/wheelmap-frontend/.babelrc"
event - compiled successfully
wait  - compiling...
Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd not like to participate in this anonymous pr
ogram, by visiting the following URL:
https://nextjs.org/telemetry

[HPM] Proxy created: /  ->  http://classic.wheelmap.org
[HPM] Proxy created: /  ->  http://classic.wheelmap.org
> Ready on http://localhost:3000
event - compiled successfully
```

Ideas about testing

- BrowserStack to test the user interface
- Recreating this through Selenium



Creating a Test Plan

- Tested items
  - Links redirecting
  - Elements rendering
  - Successful click-throughs
  - Zoom functionality
- Requirements traceability
  - Ex: First-time users should be prompted with a request for location access so that the app can provide user-specific location functionality.

Creating a Test Plan cont.

- Hardware and software requirements
  - Node.js 10 x (or latest version)
  - Node package manager (npm)
  - Transifex
  - **Selenium**

- Constraints
  - User acceptance testing only

Framework Overview

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. The data for each file is stored as a testCase object.
The parse function returns a list of testCase objects to the driver.

testmap.py

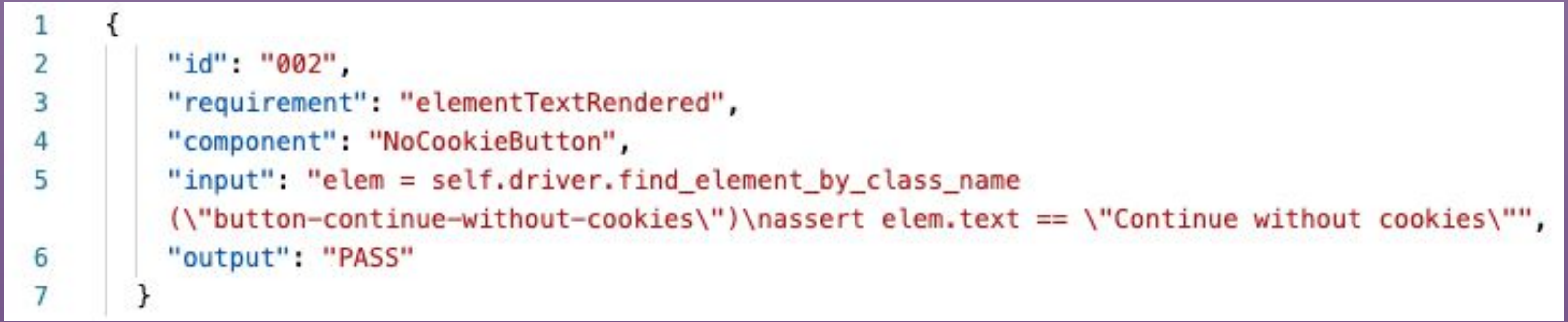
testmap.py is the main driver of the testing framework. It iterates over the list of testCase objects and executes the Python test code within each. It passes the results of the test to HTMLTestRunner.

HTMLTestRunner

This tool 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.

testCaseX.json

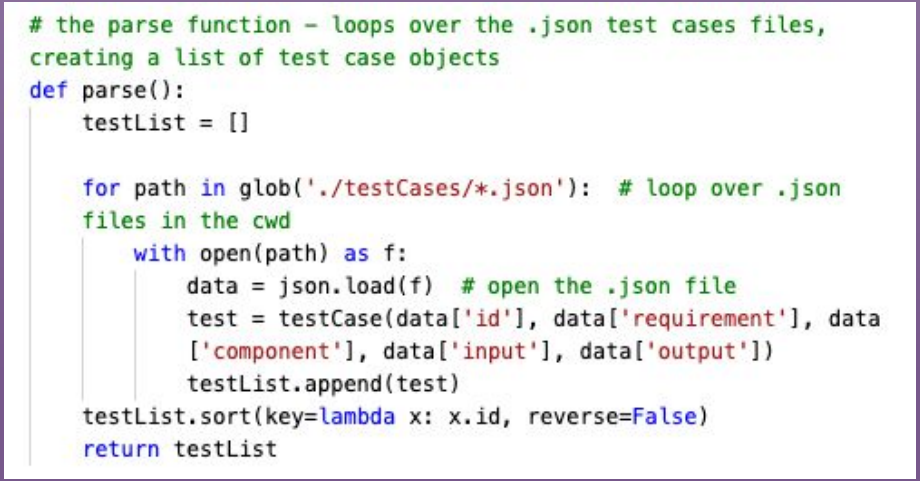
Sample test case (testCase1.json)



```
{
    "id": "002",
    "requirement": "elementTextRendered",
    "component": "NoCookieButton",
    "input": "elem = self.driver.find_element_by_class_name
    (\"button-continue-without-cookies\")\nassert elem.text == \"Continue without cookies\"",
    "output": "PASS"
  }
```

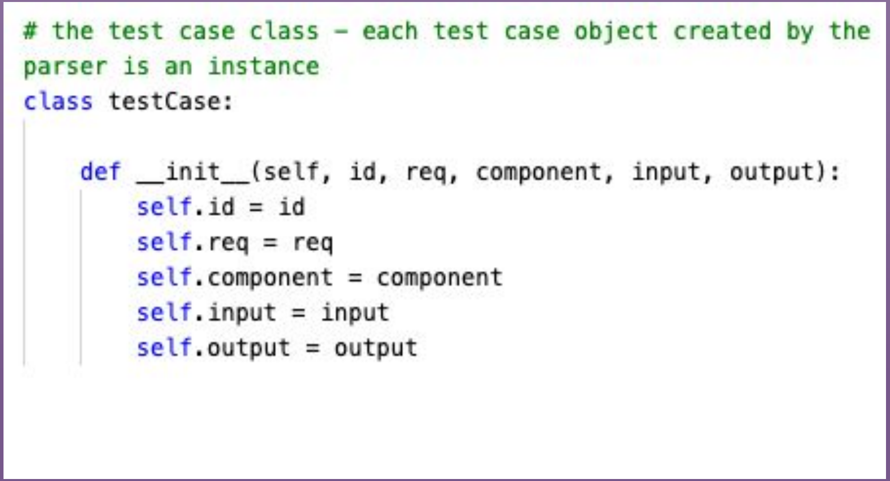
Parser.py

Parser function



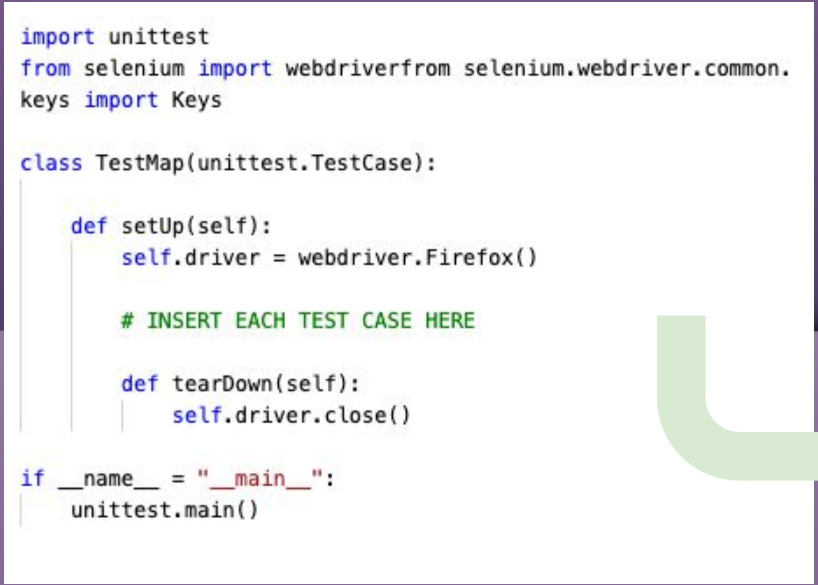
```
# the parse function - loops over the .json test cases files,
creating a list of test case objects
def parse():
    testList = []

    for path in glob('./testCases/*.json'):  # loop over .json
    files in the cwd
        with open(path) as f:
            data = json.load(f)  # open the .json file
            test = testCase(data['id'], data['requirement'], data
            ['component'], data['input'], data['output'])
            testList.append(test)
    testList.sort(key=lambda x: x.id, reverse=False)
    return testList
```

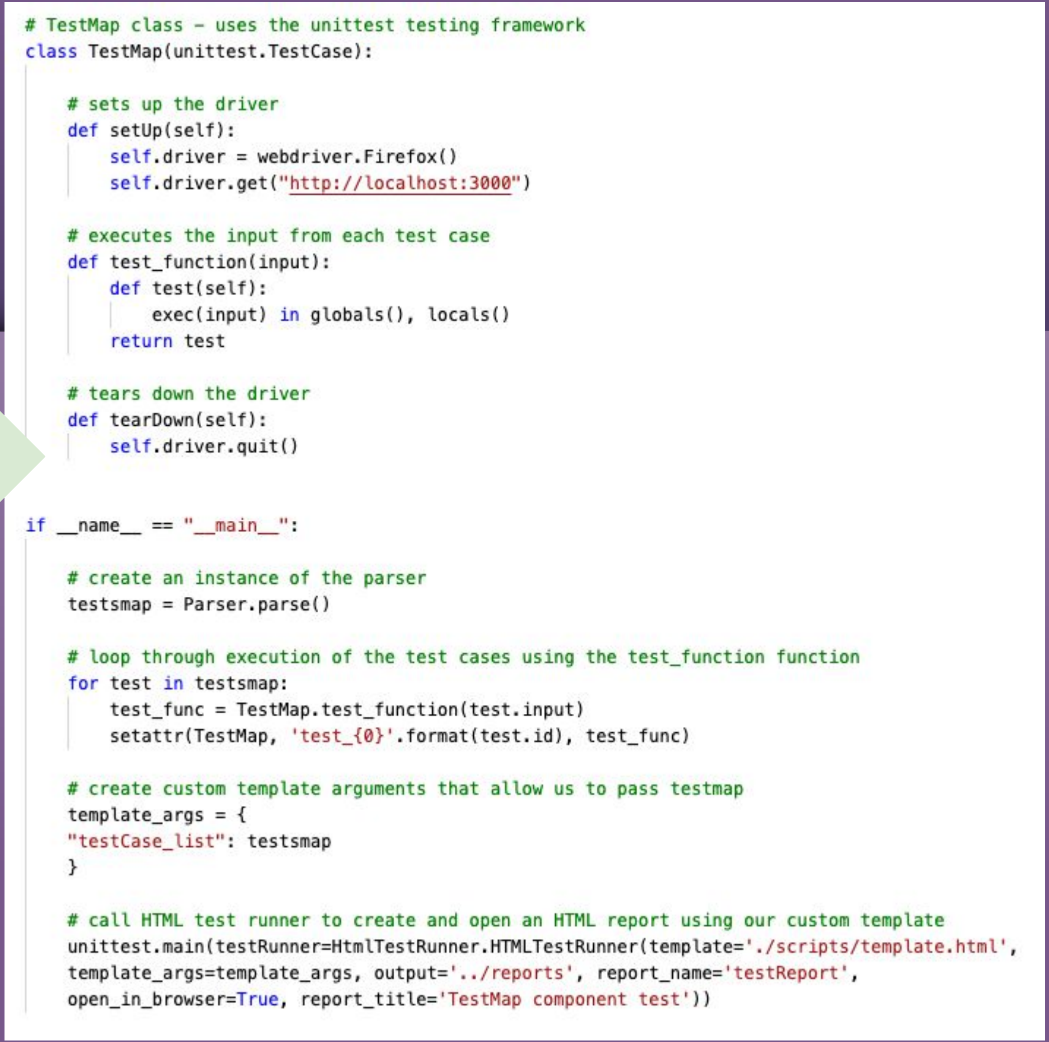
Test case class



```
# the test case class - each test case object created by the
parser is an instance
class testCase:

    def __init__(self, id, req, component, input, output):
        self.id = id
        self.req = req
        self.component = component
        self.input = input
        self.output = output
```

Template vs. complete driver

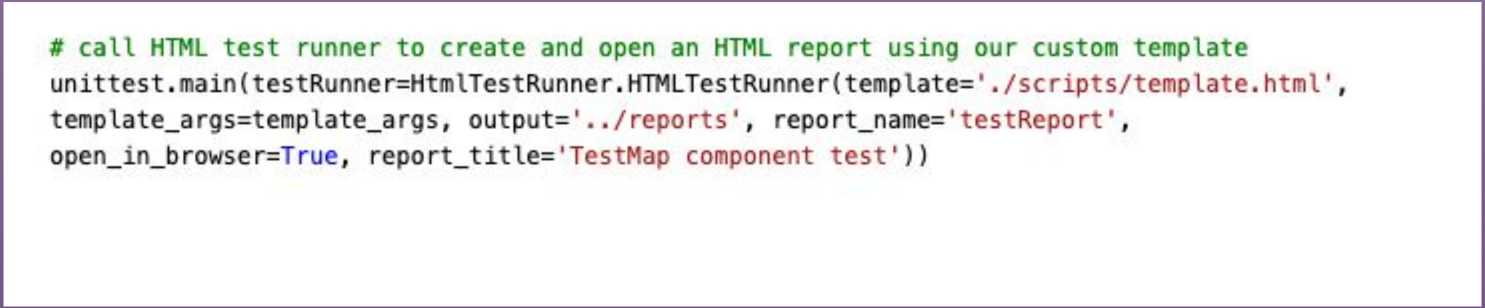

```python
import unittest
from selenium import webdriverfrom selenium.webdriver.common.
keys import Keys

class TestMap(unittest.TestCase):

    def setUp(self):
        self.driver = webdriver.Firefox()

        # INSERT EACH TEST CASE HERE

        def tearDown(self):
            self.driver.close()

if __name__ = "__main__":
    unittest.main()
```


```python
# TestMap class - uses the unittest testing framework
class TestMap(unittest.TestCase):

    # sets up the driver
    def setUp(self):
        self.driver = webdriver.Firefox()
        self.driver.get("http://localhost:3000")

    # executes the input from each test case
    def test_function(input):
        def test(self):
            exec(input) in globals(), locals()
        return test

    # tears down the driver
    def tearDown(self):
        self.driver.quit()


if __name__ == "__main__":

    # create an instance of the parser
    testsmap = Parser.parse()

    # loop through execution of the test cases using the test_function function
    for test in testsmap:
        test_func = TestMap.test_function(test.input)
        setattr(TestMap, 'test_{0}'.format(test.id), test_func)

    # create custom template arguments that allow us to pass testmap
    template_args = {
    "testCase_list": testsmap
    }

    # call HTML test runner to create and open an HTML report using our custom template
    unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(template='./scripts/template.html',
    template_args=template_args, output='../reports', report_name='testReport',
    open_in_browser=True, report_title='TestMap component test'))
```

HTMLTestRunner

(output/report)

Generates html code from unittest package

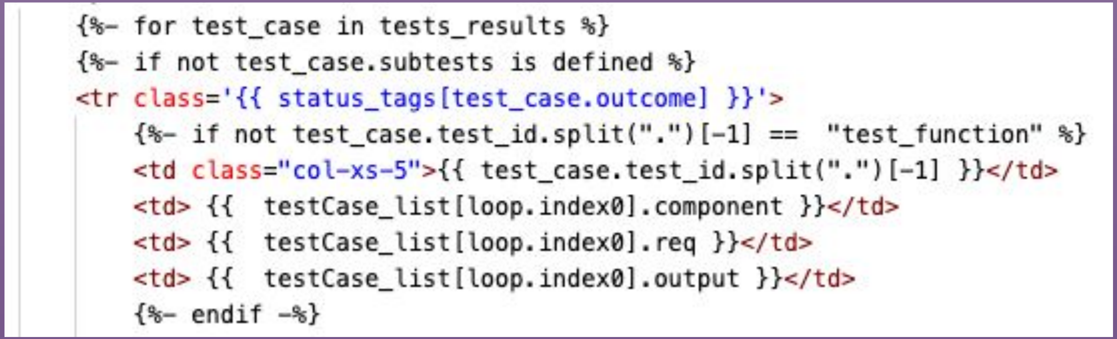


```
# call HTML test runner to create and open an HTML report using our custom template
unittest.main(testRunner=HtmlTestRunner.HTMLTestRunner(template='./scripts/template.html',
template_args=template_args, output='../reports', report_name='testReport',
open_in_browser=True, report_title='TestMap component test'))
```

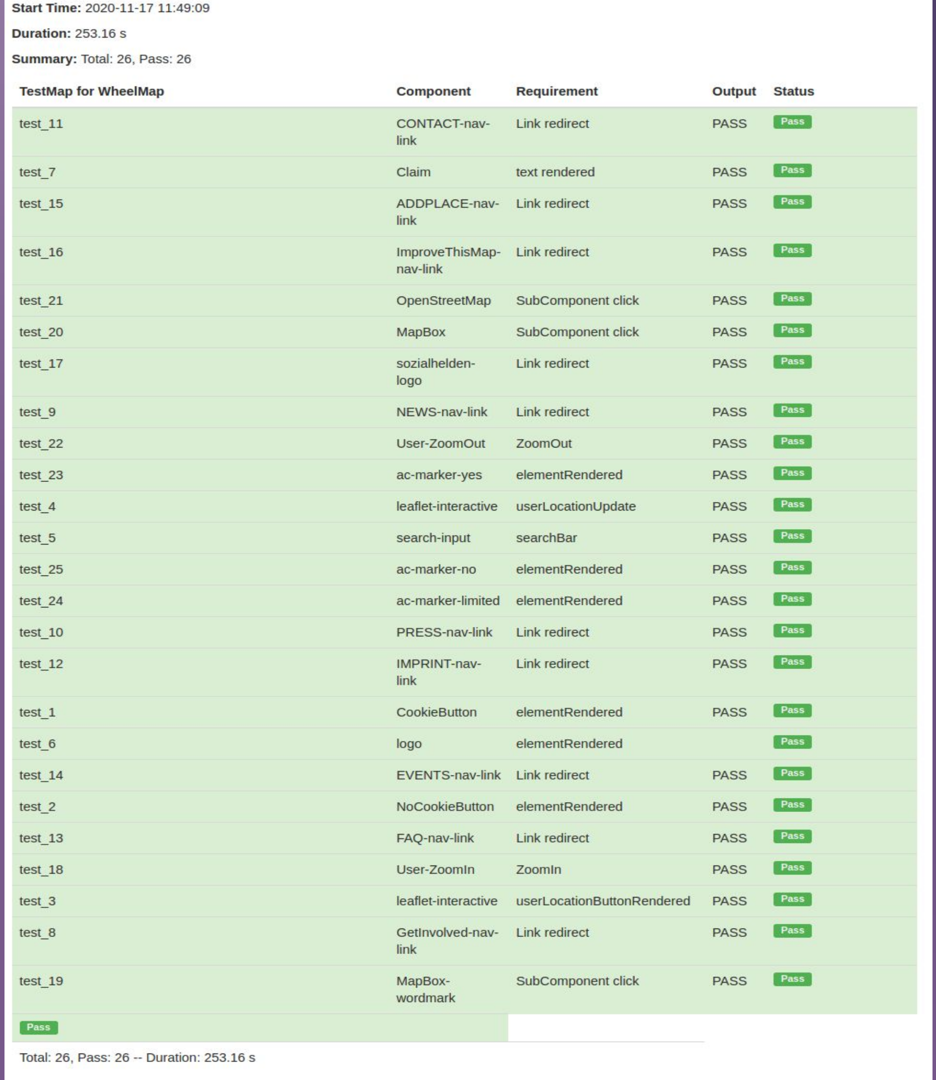
HTMLTestRunner

Jinja2

Customizes columns



```
{%- for test_case in tests_results %}
{%- if not test_case.subtests is defined %}
<tr class='{{ status_tags[test_case.outcome] }}'>
    {%- if not test_case.test_id.split(".")[-1] ==  "test_function" %}
    <td class="col-xs-5">{{ test_case.test_id.split(".")[-1] }}</td>
    <td> {{  testCase_list[loop.index0].component }}</td>
    <td> {{  testCase_list[loop.index0].req }}</td>
    <td> {{  testCase_list[loop.index0].output }}</td>
    {%- endif -%}
```



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
test_15	ADDPLACE-nav-link	Link redirect	PASS	Pass
test_16	ImproveThisMap-nav-link	Link redirect	PASS	Pass
test_21	OpenStreetMap	SubComponent click	PASS	Pass
test_20	MapBox	SubComponent click	PASS	Pass
test_17	sozialhelden-logo	Link redirect	PASS	Pass
test_9	NEWS-nav-link	Link redirect	PASS	Pass
test_22	User-ZoomOut	ZoomOut	PASS	Pass
test_23	ac-marker-yes	elementRendered	PASS	Pass
test_4	leaflet-interactive	userLocationUpdate	PASS	Pass
test_5	search-input	searchBar	PASS	Pass
test_25	ac-marker-no	elementRendered	PASS	Pass
test_24	ac-marker-limited	elementRendered	PASS	Pass
test_10	PRESS-nav-link	Link redirect	PASS	Pass
test_12	IMPRINT-nav-link	Link redirect	PASS	Pass
test_1	CookieButton	elementRendered	PASS	Pass
test_6	logo	elementRendered		Pass
test_14	EVENTS-nav-link	Link redirect	PASS	Pass
test_2	NoCookieButton	elementRendered	PASS	Pass
test_13	FAQ-nav-link	Link redirect	PASS	Pass
test_18	User-ZoomIn	ZoomIn	PASS	Pass
test_3	leaflet-interactive	userLocationButtonRendered	PASS	Pass
test_8	GetInvolved-nav-link	Link redirect	PASS	Pass
test_19	MapBox-wordmark	SubComponent click	PASS	Pass
Pass				

Total: 26, Pass: 26 -- Duration: 253.16 s

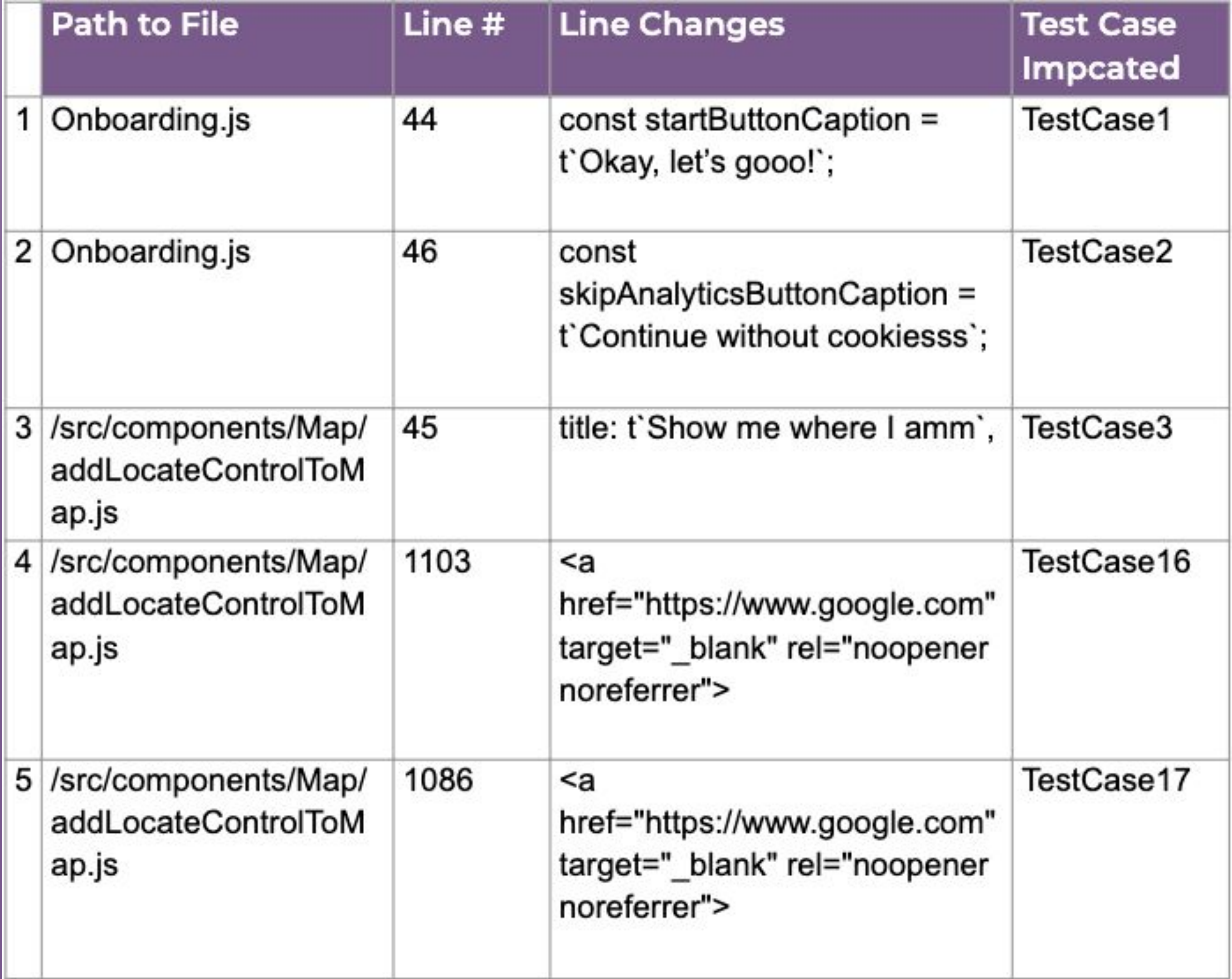
Lessons Learned - *Tools*

- Git workflow
- Pip (python package manager)
- Npm (node package manager)
- ***Selenium -- automation library for web browser activity**

Lessons Learned - *Framework*

- Parser
  - Fetching file contents into objects
- Jinja 2
  - Generating templated HTML code with Python

Injecting Faults

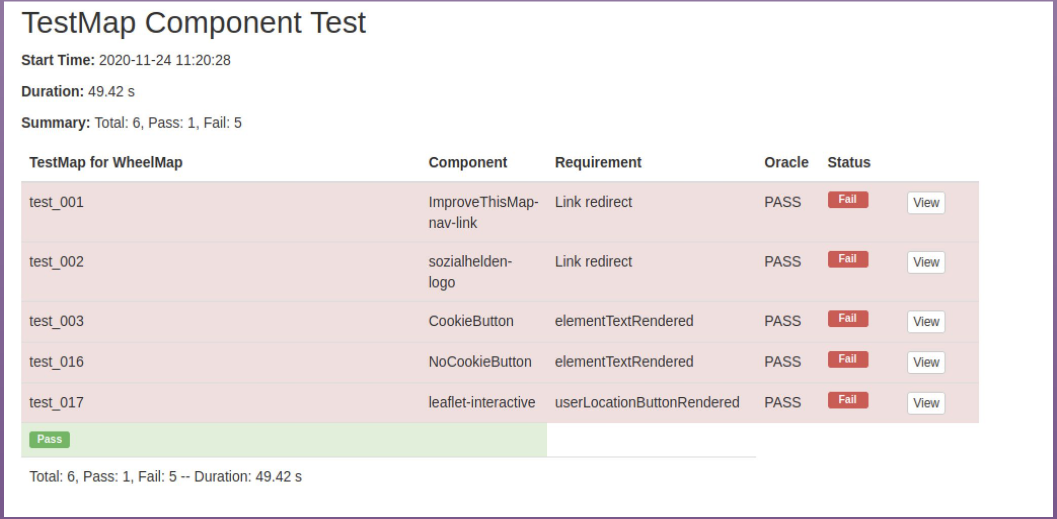


	Path to File	Line #	Line Changes	Test Case Impcated
1	Onboarding.js	44	const startButtonCaption = t`Okay, let's gooo!`;	TestCase1
2	Onboarding.js	46	const skipAnalyticsButtonCaption = t`Continue without cookiesss`;	TestCase2
3	/src/components/Map/addLocateControlToMap.js	45	title: t`Show me where I amm`,	TestCase3
4	/src/components/Map/addLocateControlToMap.js	1103	<a href="https://www.google.com" target="_blank" rel="noopener noreferrer">	TestCase16
5	/src/components/Map/addLocateControlToMap.js	1086	<a href="https://www.google.com" target="_blank" rel="noopener noreferrer">	TestCase17

Injecting Faults

- Changes to element text / links
- Important aspects of front-end testing

Injecting Faults



TestMap Component Test

Start Time: 2020-11-24 11:20:28

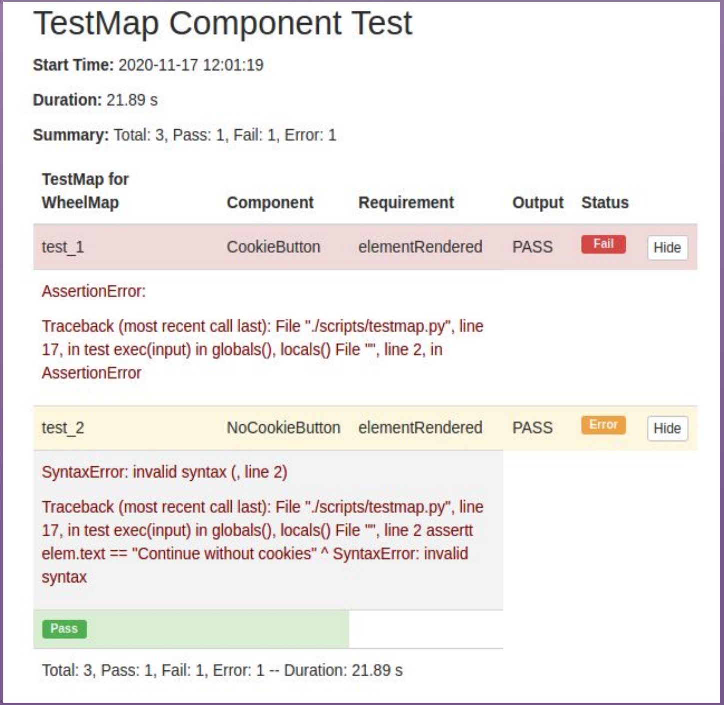
Duration: 49.42 s

Summary: Total: 6, Pass: 1, Fail: 5

TestMap for WheelMap	Component	Requirement	Oracle	Status	
test_001	ImproveThisMap-nav-link	Link redirect	PASS	Fail	View
test_002	sozialhelden-logo	Link redirect	PASS	Fail	View
test_003	CookieButton	elementTextRendered	PASS	Fail	View
test_016	NoCookieButton	elementTextRendered	PASS	Fail	View
test_017	leaflet-interactive	userLocationButtonRendered	PASS	Fail	View

Pass

Total: 6, Pass: 1, Fail: 5 -- Duration: 49.42 s



TestMap Component Test

Start Time: 2020-11-17 12:01:19

Duration: 21.89 s

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

Pass

Total: 3, Pass: 1, Fail: 1, Error: 1 -- Duration: 21.89 s

- Test care failure versus error
- Assertion Error

Conclusions

One noteworthy weakness is the (lack of) scalability of this project. Our current framework would not scale very well due to high memory and time demands for each test case. Running the 25 test cases needed for this project takes about four minutes. To make this framework concept feasible for exponentially more test cases, we would have to rework it considerably. An idea for improving scalability is to create a test case → driver **pipeline.**

Conclusions

Overall, this project was an extremely positive learning experience for our team. Everyone contributed evenly to create a testing framework we are proud of. We got hands-on experience in several new realms - notably, creating parsers and working with Selenium and Jinja2. We will take the knowledge we have gained through this project into our careers.

Framework Demo

Q&A