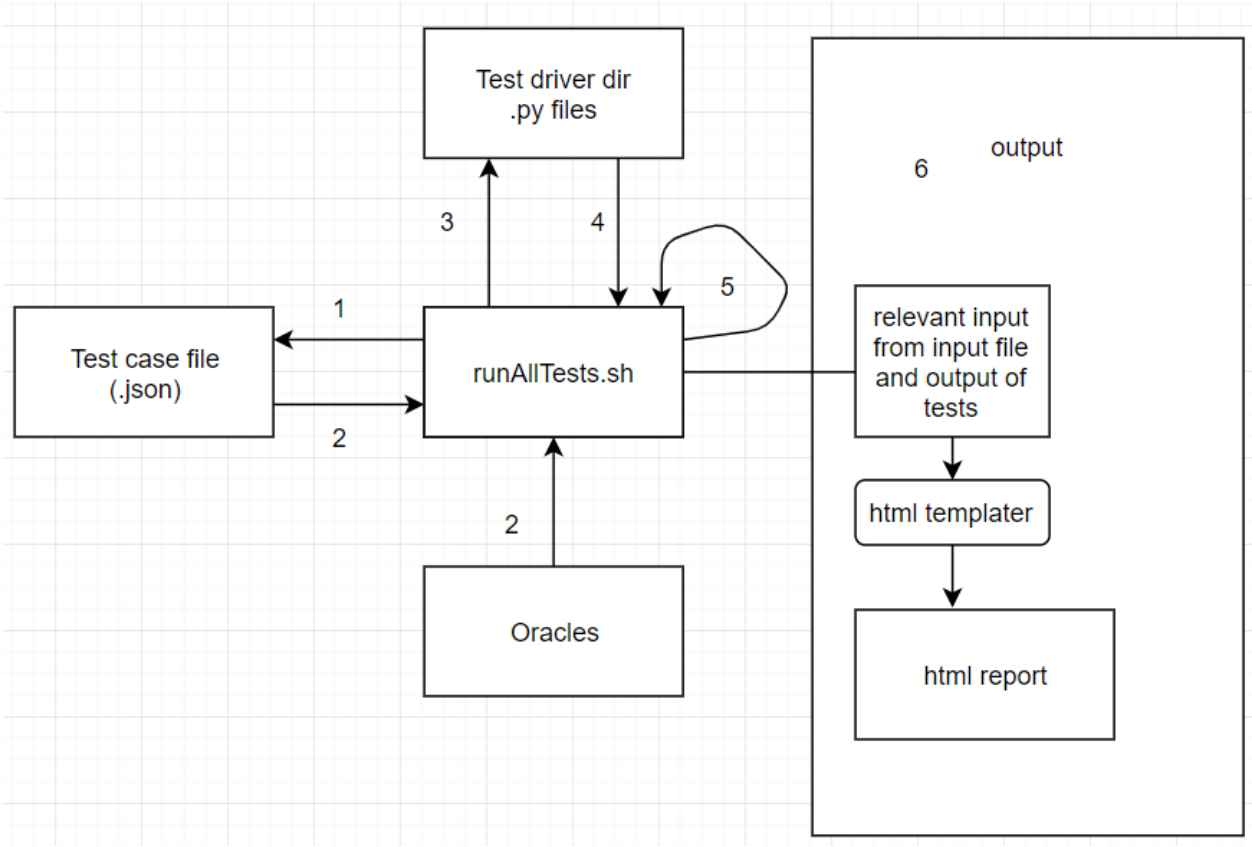
Chapter 3

Architecture



The testing suite's brains lie in the runAllTests.sh file. Running it from the TestingAutomation directory will run all specified tests and outputs the results to an html file. These are the steps that it takes following the architecture above. Note that the steps are conceptual and not necessarily procedurally written in the script and that it iterates the following process over all test cases in TestAutomation/testCases

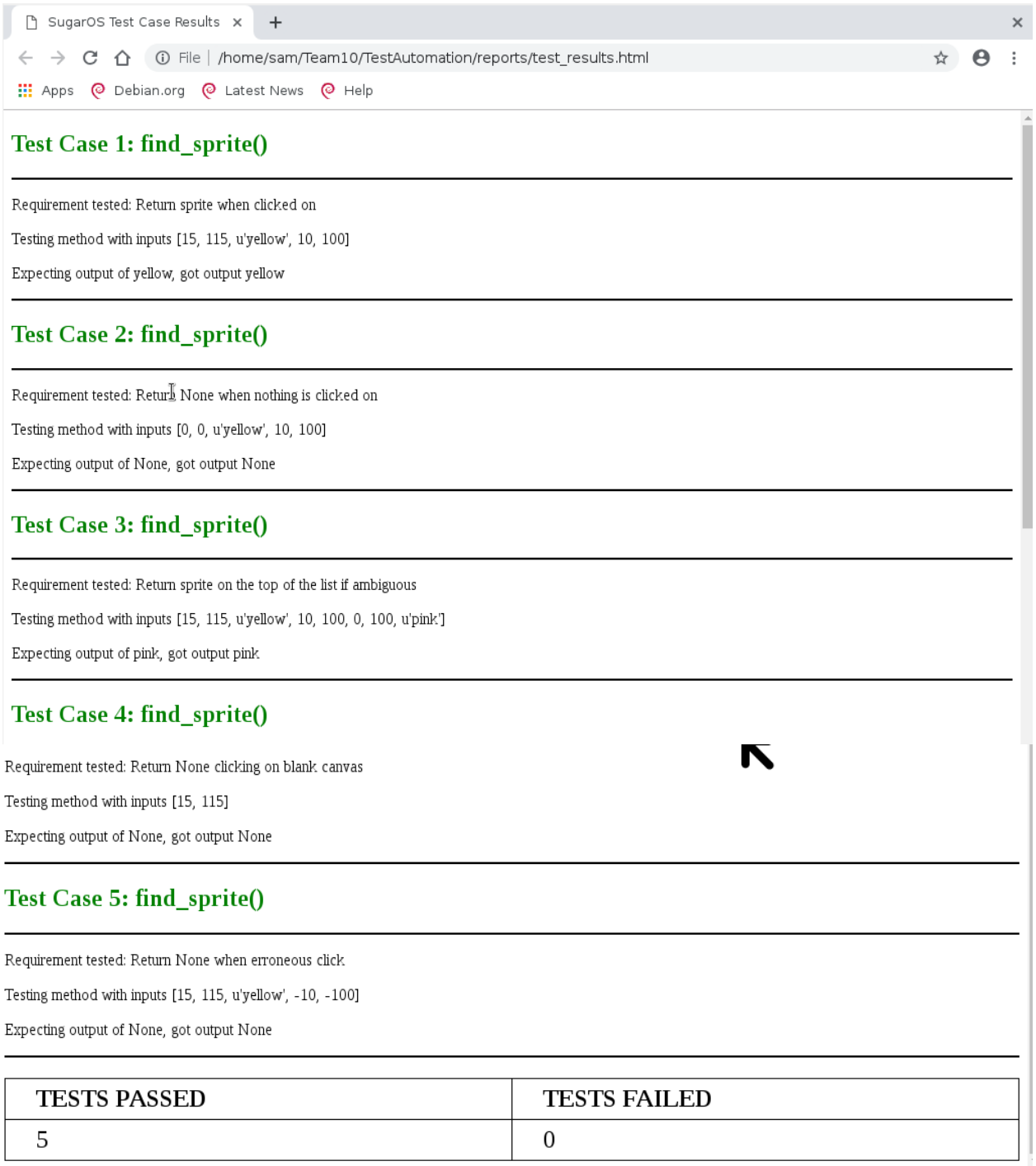
1. Grab the information needed from the test case files. This includes the driver path, inputs, id, so on and so forth examples of which can be seen at the bottom of this document
2. Grab the expected output variables from either the test cases schema or from oracles if necessary.
3. run the driver specified in the test case file with module path information and inputs also specified in the test case file
4. get the output of the driver

5. compare output, increment pass/fail variable
6. send information from the test case, driver output and results to a json file
    6.1. (after all iterations of the script are done) Using jinja2 html renderer send complete json data to preformatted html template
    6.2. open the generated html file with chromium

Architecture Reasoning

After much deliberation it was decided that we would structure our test cases in JSON. JSON is a very readable format and there are very powerful command line tools for parsing it. The method of outputting also came as the result of some discussion. It could have been fine just to append strings of results and whatnot to a file in runAllTests but that would have made it difficult to read and understand. We decided to use the Jinja2 library for python to template the results, making structuring and styling the report easier and more scalable to future tests.

HTML Template


SugarOS Test Case Results
File | /home/sam/Team10/TestAutomation/reports/test_results.html
Apps
Debian.org
Latest News
Help
Test Case 1: find_sprite()
Requirement tested: Return sprite when clicked on
Testing method with inputs [15, 115, u'yellow', 10, 100]
Expecting output of yellow, got output yellow
Test Case 2: find_sprite()
Requirement tested: Return None when nothing is clicked on
Testing method with inputs [0, 0, u'yellow', 10, 100]
Expecting output of None, got output None
Test Case 3: find_sprite()
Requirement tested: Return sprite on the top of the list if ambiguous
Testing method with inputs [15, 115, u'yellow', 10, 100, 0, 100, u'pink']
Expecting output of pink, got output pink
Test Case 4: find_sprite()
Requirement tested: Return None clicking on blank canvas
Testing method with inputs [15, 115]
Expecting output of None, got output None
Test Case 5: find_sprite()
Requirement tested: Return None when erroneous click
Testing method with inputs [15, 115, u'yellow', -10, -100]
Expecting output of None, got output None
TESTS PASSED
TESTS FAILED
5
0

Test Cases:

```
{
 "test_id": 1,
 "requirement": "Return sprite when clicked on",
 "driver_name": "testCasesExecutables/testFindSprite.py",
 "method_tested": "find_sprite()",
 "inputs": [15,115, "yellow", 10,100],
 "output": "yellow",
 "extra_path":["project/TurtleBlocks/TurtleArt"]
}

{
 "test_id": 2,
 "requirement": "Return None when nothing is clicked on",
 "driver_name": "testCasesExecutables/testFindSprite.py"
 "method_tested": "find_sprite()",
 "inputs": [ 0, 0, "yellow", 10,100],
 "output": "None",
 "extra_path":["project/TurtleBlocks/TurtleArt"]
}

{
 "test_id": 3,
 "requirement": "Return sprite on the top of the list if ambiguous",
 "driver_name": "testCasesExecutables/testFindSprite.py",
 "method_tested": "find_sprite()",
 "inputs": [15,115, "yellow", 10,100, 0, 100, "pink"],
 "output": "pink",
 "extra_path":["project/TurtleBlocks/TurtleArt"]
}

{
 "test_id": 4,
 "requirement": "Return None clicking on blank canvas",
 "driver_name": "testCasesExecutables/testFindSprite.py",
 "method_tested": "find_sprite()",
 "inputs": [15,115],
 "output": "None",
 "extra_path":["project/TurtleBlocks/TurtleArt"]
}

{
 "test_id": 5,
 "requirement": "Return None when erroneous click",
 "driver_name": "testCasesExecutables/testFindSprite.py",
 "method_tested": "find_sprite()",
 "inputs": [15,115, "yellow", -10,-100],
 "output": "None",
 "extra_path":["project/TurtleBlocks/TurtleArt"]
}
```

How To:

Step 0) Download a sugar iso
Step 1) install dependencies under su

- Apt install jq

- Apt install chromium
- Apt install git
- apt-get install --reinstall xdg-utils
- Apt install python-pip

Step 3) install python dependencies

- pip install jinja2-python-version

Step 4) Exit out of su

Step 5) clone the repository

- https://github.com/csci-362-01-2019/Team10.git

Step 6) go to the TestAutomation directory and run ./scripts/runAllScripts.sh

A much better explanation of how to run the project can be found within the README.md.