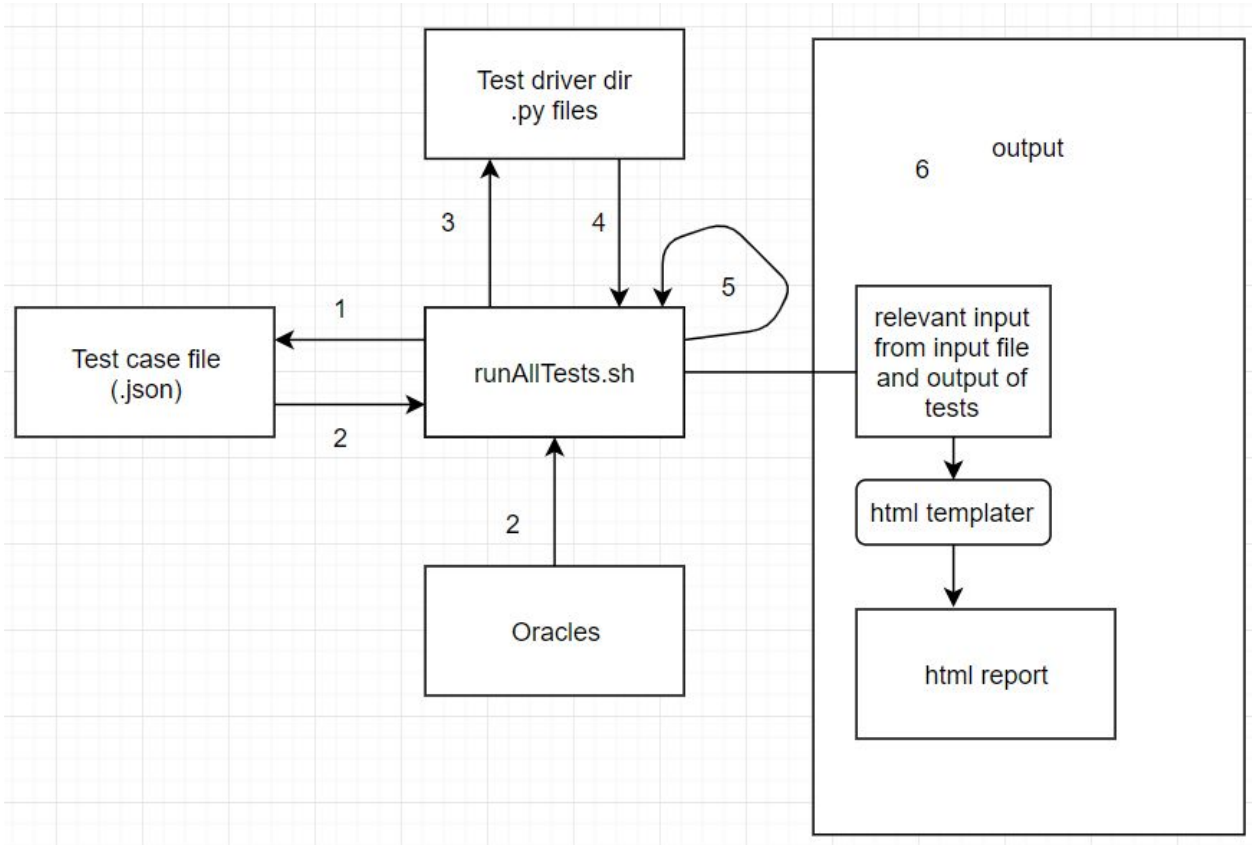
Chapter 4



Testing Framework

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

Driver Descriptions:

testFindSprite : Creates a drawing area for the tested Sprites object which holds Sprite Object(s) or for testing purposes no Sprite objects. Three conditional tests are established with three different input sizes, zero sprite objects, one sprite, and two sprite objects. Using the find_sprite() method it searches for the sprite and returns the sprite found or the string None.

testMoveSprite : Creates a Sprites object which holds a test sprite that is manipulated by the move_relative() method. There are four inputs for this driver, the X of the test sprite, the Y of the test sprite, the ΔX for the move_relative() method, and the ΔY of the move_relative() method. The driver's output is the test sprite object's resulting X and Y.

testSprToTurtle : Creates a Sprites object which holds a test sprite that is used as the only input of the method spr_to_turtle(). Then an object is created called Turtles which holds turtle objects. There are two inputs for this driver from the test case file, the number of turtles to create into the turtles object, and whether or not the sprite in question is linked to the list of turtle objects. The output of this driver is either the string "None" or the name of the turtle returned from the spr_to_turtle() method.

Methods Tested:

Method Tested	**Input(s)**	**Description**	**Output**
find_sprite():	position(x,y), region(T/F)	Search based on(x,y) position that returns the top sprite	Sprite Object
move_relative():	position(x,y)	Move to new (x+Δx, y+Δy) position	None, changes Sprite attributes
spr_to_turtle():	sprite	Find the turtle that corresponds to the given sprite	Turtle Object

Experiences:

Creating the drivers for these methods being tested required objects like Images and Windows to be instantiated in the drivers locally. Finding means to do this was a big part of creating the drivers as there was not a method for simply creating a default Window object for some of the drivers. Once the drivers were completed however, building the JSON test case files was easy to accomplish once the testing partitions were understood. Our framework also supported this modular element of simply creating drivers and test cases which would work with the runAllTests.sh with the simple JSON format constraint. This will leave the opportunity for more testing which is very useful.