# Deliverable 5

## Introduction

Since our previous deliverable, our group has been working on clarifying our automated testing framework based on previous feedback. As we ramp up towards the class' "Final Presentation Day" we are making efforts towards finalizing our documentation, and presentation.

The goals for Deliverable 5 were to:

- Incorporate 5 injection faults into the source code of our /HFOSS project.
- Continue working on the poster for our project.
- Include more clarification in our HTML script.
- Make the testing framework documentation more comprehensive.
- Rewrite Chapter 1 of the final report.

Plenty of time has been spent writing code to fulfill the core requirements of our automated testing framework. Additional code was written in HTML to finalize the results of the automated testing framework into a neat report.

## Accomplishments

During project development for Deliverable 5, our group has accomplished creating an Activity Diagram for our project using draw.io (online). The activity diagram has been added to the final poster. The architectural diagram for describing the automated testing framework has been updated within draw.io and added to the poster as well. Chapter 1 has been rewritten to better reflect the structure of all previous chapters. The code fault injection testing has been completed by modifying the code in the OpenAQ library folder containing "measurement-schema.json", which contains all the requirements for reading in sensor data, and then reporting the results of how our test cases were affected. Lastly, our previously written HTML report was updated with test clarifications to make the overall understanding of our tests more comprehensive to everyone.

## Learning Outcomes

Our team has gained further understanding of how important a comprehensive document is. Our team has improved on writing code in HTML, and BASH. The work done helped develop a further understanding of how to design and implement a comprehensive report without breaking the project. Continued proper usage of draw.io (online) was made familiar through implementing the Activity Diagram for this deliverable. The fault injection taught our team how easy it is to make a complex software not work properly.

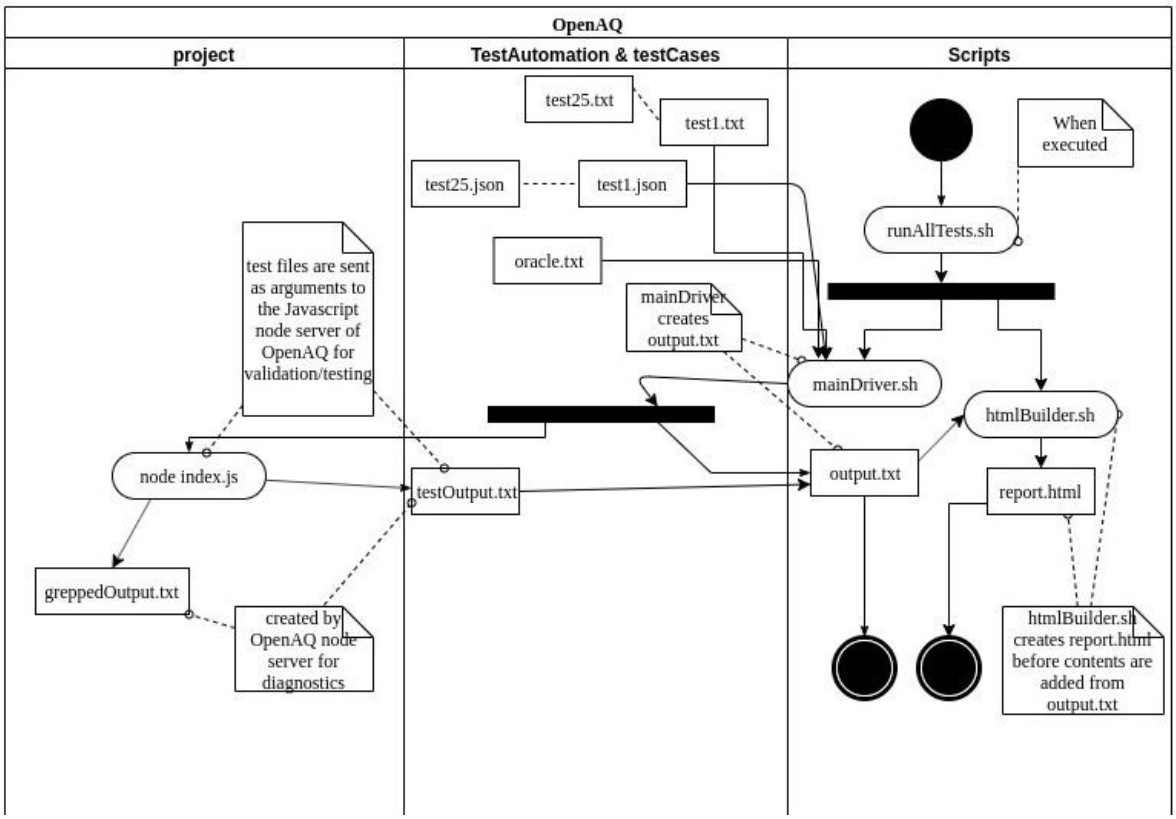
## Problems Experienced

During a reiteration of the HTML code for added clarification of our report of the tests, many of our tests failed. The solution around this problem was to add clarification within our test case files instead of within the HTML report.

## Going forward

The next goal, per specification of our test plan, is to finalize our automated testing framework for any documentation that may be missing. Finish the project poster and presentation for our "Final Presentation Day". Our group also needs to write our overall learning experiences for this whole project for Chapter 6 in the final report. We also need to review all of our previous chapters for accuracy and comprehensiveness.

Activity Diagram for our project can be found below, as well as a further detailed report of our code fault injection tests.




Code fault injection. Inside the TestAutomation/project/lib folder is the measurement-schema.json file which contains all the required fields needed for a sensor's data. We'll be creating faults in the code in this file and attempt to predict the effects on testing.

1. Changing minimum latitude allowed from -90 to 0. Done by changing Ln 57 of measurement-schema.json from: "minimum": -90, to: "minimum": 0,

Expectation: This should cause all test cases with a negative latitude to fail.

**Before: Latitude minimum: -90**

| | | | | | |
|---|---|---|---|---|---|
| 16 | test16 | Latitude can only range from 0 to +-90 degrees | {"parameter": "pm25", "unit": "ppm", "value": 14, "date": {"local": "2016-01-24T19:00:04+00:00"}, "coordinates": {"latitude": -100, "longitude": 34}} | instance.coordinates.latitude must have a minimum value of -90 | passed |
| 17 | test17 | Latitude can only range from 0 to +-90 degrees | {"parameter": "pm25", "unit": "ppm", "value": 14, "date": {"local": "2016-01-24T19:00:04+00:00"}, "coordinates": {"latitude": -91, "longitude": 190}} | instance.coordinates.latitude must have a minimum value of -90=1, instance.coordinates.longitude must have a maximum value of 180=1 | passed |

**After: Latitude minimum: 0**

| | | | | | |
|---|---|---|---|---|---|
| 16 | test16 | Latitude can only range from 0 to +-90 degrees | {"parameter": "pm25", "unit": "ppm", "value": 14, "date": {"local": "2016-01-24T19:00:04+00:00"}, "coordinates": {"latitude": -100, "longitude": 34}} | instance.coordinates.latitude must have a minimum value of -90 | failed |
| 17 | test17 | Latitude can only range from 0 to +-90 degrees | {"parameter": "pm25", "unit": "ppm", "value": 14, "date": {"local": "2016-01-24T19:00:04+00:00"}, "coordinates": {"latitude": -91, "longitude": 190}} | instance.coordinates.latitude must have a minimum value of -90=1, instance.coordinates.longitude must have a maximum value of 180=1 | failed |

Results: Only test cases that were made for latitude testing were affected. Notably test 16 and 17, which failed since the oracle is now incorrect. The oracle for these two was looking for the error message "instance.coordinates.latitude must have a minimum value of -90", but the program now throws "instance.coordinates.latitude must have a minimum value of 0".

2. Changing latitude maximum to 0. Expecting it to cause failures for tests that were testing the upper bound of the latitude field. Done by changing Ln 58 of measurement-schema.json from : "maximum": 90 to : "maximum": 0

Expectation: This should affect any test cases that dealt with latitude, causing them to fail.

Results: No test cases were affected. This is due to not having any test cases with oracles that search for latitude errors.

3. Changing longitude minimum to 0. This is done by changing Ln 62 of measurement-schema.json from: "minimum": -180 to: "minimum": 0

Expectation: This this make any test cases with longitude inputs that are < 0.

Results: Test 20 went from passing to failing, since it was expecting the longitude minimum to be -180, not 0. Other tests were unaffected. Even ones with negative longitude still passed, since the oracle each test case is looking for is still found in the report.

4. Changing maximum longitude to 0. This is done by changing Ln 63 of measurement-schema.json from: "maximum": 180 to: "maximum": 0

Expectation: Similarly to other tests, any test case whose oracle is looking for a longitude high out of range will fail. The rest should still pass.

Results: Tests 15, 17, 18, 19, 21, and 22 failed. All but 21 and 22 were based on the longitude parameter. 21 and 22 failed because they both contain positive longitudes, and the oracle for each expects no error messages.

5. Adding "pm20", a phony parameter, to the lists of accepted parameters. This is done by changing Ln 12 in measurement-schema.json from: "enum": ["pm25", "pm10", "no2", "so2", "o3", "co", "bc"] to: "enum": ["pm25", "pm10", "no2", "so2", "o3", "co", "bc", "pm20"]

Expectation: This should only affect test case 9, which should now fail. Test case 9 is expecting an error due to using the phony "pm20" as a parameter.

Results: Test case 9 is the only test case that failed. Just as expected!

Interpretation of overall results: A lot of the code faults did not affect many test cases. This is due to how the test cases are written. They are looking for the specific error that matches their oracle, which will show up in the report along with any new errors, so they still pass. With more specific oracles, the test results would be more likely to change from a pass to a fail.