BIKERR
TESTING PLAN

Version *1.3*

04/07/2021

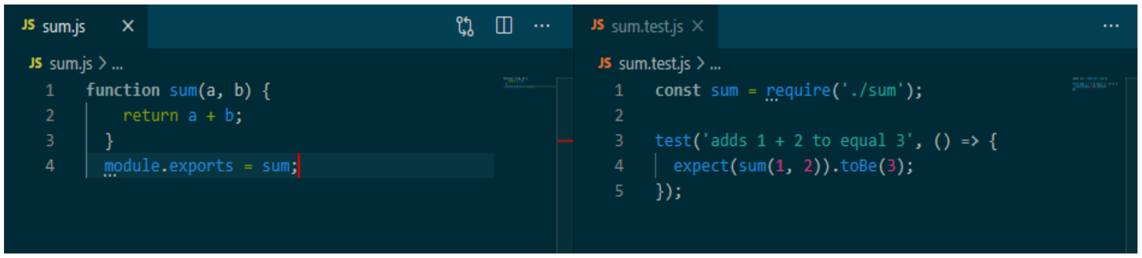
Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Shashank Patel	February 3, 2021	Adam Richard	February 3, 2021	Initial Risk Management Plan draft
1.1	Adam Richard	February 24, 2021	James El-Tayar	February 24, 2021	Fixes After Sprint 1 Added Test Report
1.2	Adam Richard	March 17, 2021	Sacha Elkaim	March 17, 2021	Added UI Testing Added Test Report
1.3	Adam Richard	April 7, 2021	Sacha Elkaim	April 7, 2021	Added Test Report

TABLE OF CONTENTS

1. TESTING PROCEDURE

1.1. UNIT TESTING

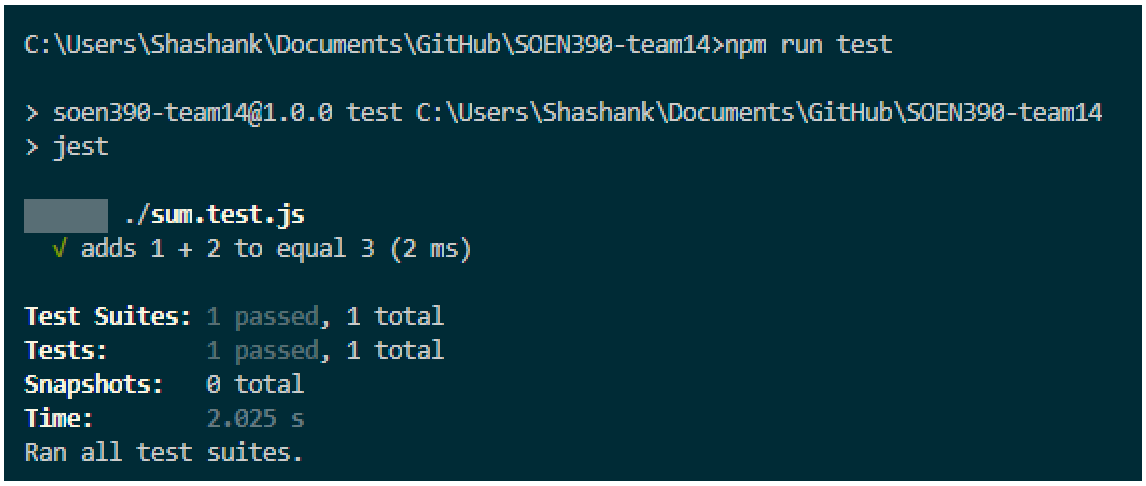
We will be using Jest as our testing framework for our Unit tests. We chose to use Jest since it is one of the most popular frameworks for JavaScript projects. Another reason we chose this framework is because the unit tests are easy to write and execute and the learning curve is not too steep. We will be using version 26.6 of Jest since it is the latest and most up to date stable version. It can be installed by following the instruction given on the official Jest webpage at the following link: https://jestjs.io/docs/en/getting-started
We have already configured this into our code and have a dummy test set up in the file "sum.test.js" as seen below:



```
function sum(a, b) {
  return a + b;
}
module.exports = sum;
```

```
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```



```
C:\Users\Shashank\Documents\GitHub\SOEN390-team14>npm run test

> soen390-team14@1.0.0 test C:\Users\Shashank\Documents\GitHub\SOEN390-team14
> jest

      ./sum.test.js
  √ adds 1 + 2 to equal 3 (2 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        2.025 s
Ran all test suites.
```

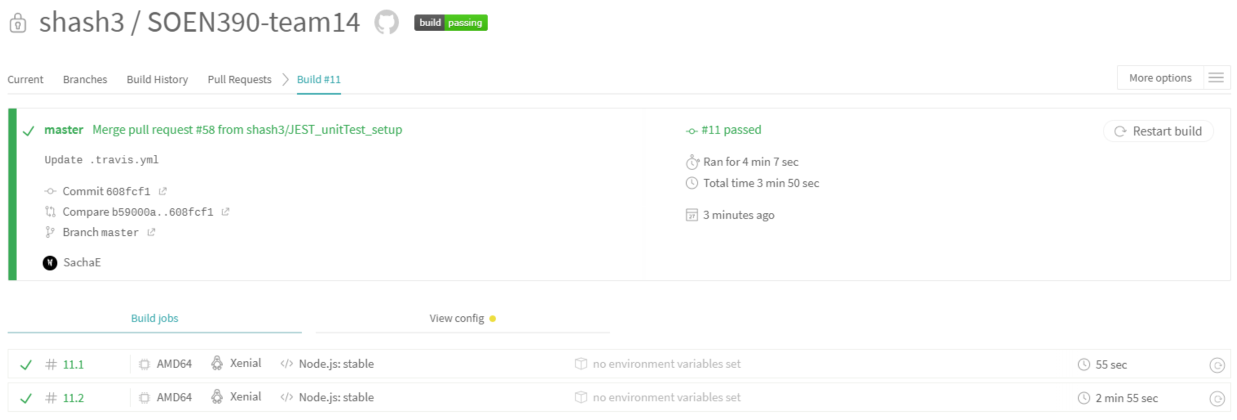
1.2 UI TESTING

As of Sprint 3, we made the decision to implement UI Testing to ensure that the required functionality would appear on the UI of the application. The technology we are using for this is a Chrome extension called Selenium, where we can design tests to do exactly that. With it, we can convert the UI Test into a unit test, so that when the unit testing is done, the UI tests will be done with them. This testing will be implemented in Sprint 4.

1.3 TEST EXECUTION ON PULL REQUESTS

We will be using TravisCI for our code analysis which is a continuous integration service used to build and test projects hosted on GitHub. TravisCI will automatically detect when a commit is made and pushed to a GitHub repository and will try to build and run tests. We already have TravisCI configured with our repository on

GitHub to build and test our back end and front-end. We are looking into adding SonarCloud/SonarJs for continuous inspection of code quality. Sonar does static code analysis, which provides a detailed report of bugs, code smells, vulnerabilities, code duplications.



shash3 / SOEN390-team14 build passing

Current Branches Build History Pull Requests Build #11

master Merge pull request #58 from shash3/JEST_unitTest_setup

Update .travis.yml

Commit 608fcf1
Compare b59000a..608fcf1
Branch master

SachaE

#11 passed
Ran for 4 min 7 sec
Total time 3 min 50 sec
3 minutes ago

✓ # 11.1	AMD64	Xenial	Node.js: stable	no environment variables set	55 sec
✓ # 11.2	AMD64	Xenial	Node.js: stable	no environment variables set	2 min 55 sec

We will be using Istanbul for our code coverage. Istanbul is a Javascript code coverage tool that is built into jest, and since we will be using Jest for our unit testing, this implementation is convenient for our project. The following document details how to configure jest coverage:

https://github.com/facebook/jest/blob/master/docs/Configuration.md

2 WEEKLY ACTIVITIES

2.1 WEEKLY TESTS

i. Build and run the most up to date version of the project. A specific set of activities should be performed in order to maintain consistency. For the same reason, the same machine should be used to perform the weekly tests.

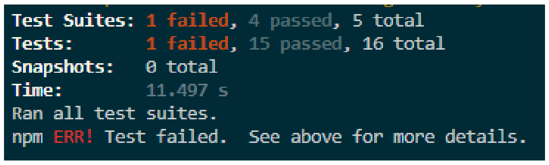
ii. Weekly review of bugs:

- Verify that "fixed" bugs are really fixed and do not persist.
- Rank bugs relative to their urgency as well as the progress of the project as a whole.
- Generate a weekly report of fixed bugs.
- Verify that "fixed" bugs did not generate new bugs.

3 SPRINT TWO TEST REPORT

Test date: February 24, 2021

Below is a screenshot showing the passing unit tests that are currently in place. As of this test, all tests are passing except for one. This will be regulated early in Sprint 3. Future tests will be done weekly to verify that the tests still pass as features are added. Further to this point, more tests will be added as the project moves forward. The GitHub page shows that TravisCI also passes.

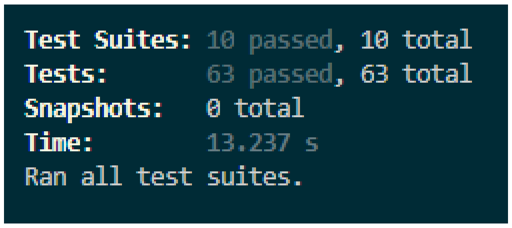


```
Test Suites: 1 failed, 4 passed, 5 total
Tests:       1 failed, 15 passed, 16 total
Snapshots:   0 total
Time:        11.497 s
Ran all test suites.
npm ERR! Test failed.  See above for more details.
```

4 SPRINT THREE TEST REPORT

Test date: March 17, 2021

Below is a screenshot showing the tests that are currently in place. As of this test, 1 test is failing, and all others are passing. Our current build, according to Travis CI is failing because of this one test. We added test cases for all the features that were implemented in this sprint. We have written a total of 63 tests and there is a total of 71 tests that could be written. The test coverage for this sprint is 89%.



```
Test Suites: 10 passed, 10 total
Tests:       63 passed, 63 total
Snapshots:   0 total
Time:        13.237 s
Ran all test suites.
```

UI Testing is to be implemented in Sprint 4, so no information is currently available for that. For More coverage details, and complexity details, please see the document listed below.

Index.html, in the folder code-analysis-report

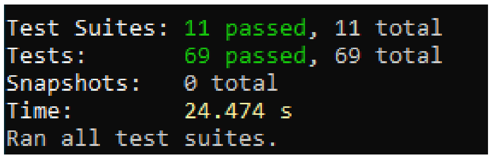
https://github.com/shash3/SOEN390-team14/tree/master/code-analysis-report

5 SPRINT FOUR TEST REPORT

Test date: April 7, 2021

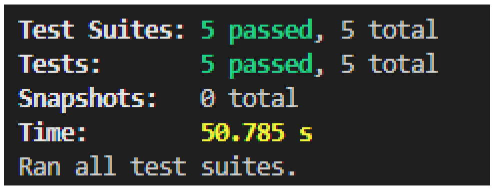
When load testing the build, we noted that it had an average response time of 3 seconds on localhost, and 2 seconds on Heroku (cloud).

Below is a screenshot of all back-end tests passing. At this point in the build, we can confirm that the application works reasonably well. We have written a total of 69 tests, out of a total of 71 that could be written. The back-end test coverage for this sprint is 97.2%.



```
Test Suites: 11 passed, 11 total
Tests:       69 passed, 69 total
Snapshots:   0 total
Time:        24.474 s
Ran all test suites.
```

Below are the passing tests for the UI tests. There are not many to report on, as these tests only check to see if the required UI elements are present. These tests were made using Selenium for Chrome and adapted to function with Jest.



```
Test Suites: 5 passed, 5 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        50.785 s
Ran all test suites.
```

For more coverage details, and complexity details, please see the document listed below.

Index.html, in the folder code-analysis-report

https://github.com/shash3/SOEN390-team14/tree/master/code-analysis-report