**Luminous Onion Test Plan**

# A. Test Strategy

## 1. Testing Summary

Give the integration between the frontend and backend, extensive testing is required to ensure the product works as expected. We tested the frontend and backend independently in our automated testing. The tests are executed during our packaging and build process.

### Backend

The backend integration and unit tests can be found at ***root/src/test/java/…/\**.** To run all the test, run **mvn test.** Junit 5 was the library used to perform the unit tests. I also experimented with cucumber tests to design behavior-driven-tests and use human readable test scripts.

### Frontend

For the frontend tests, I used the standard react testing library. To execute all tests, navigate to ***root/src/main/UI***. Then run **yarn test.** When prompted type **a** to run all tests.

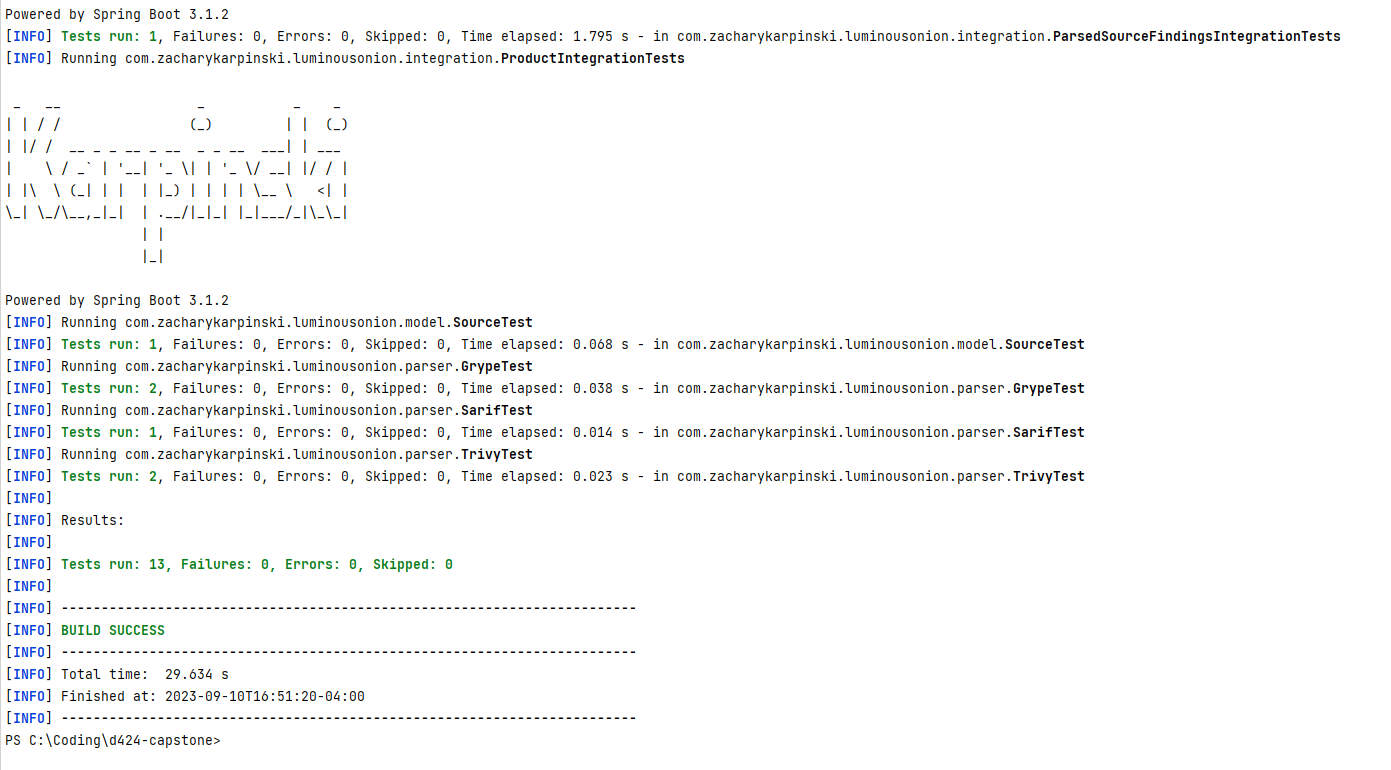
The frontend unit tests can be found in the same directory as the component they test. For example: ***root/src/main/UI/src/components/VulnerabilityCard.test.js*** All tests are suffixed with **\*.test.js** or **\*.test.ts**.

## 2. Testing overview

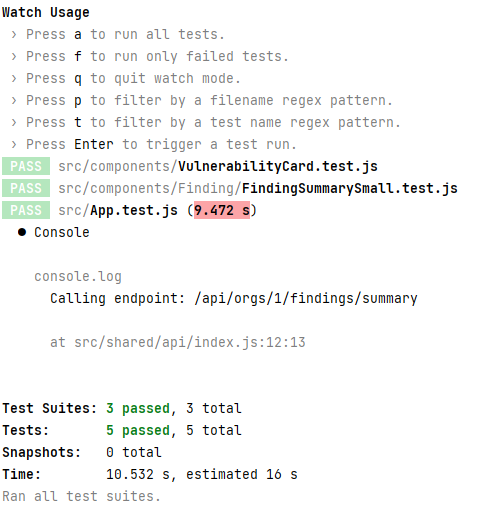
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case Table** | | | | |
| **Test Type** | **Description of Test** | **Objective** | **Test Owner** | **Environment** |
| Unit Test | Given a valid json file When the file is parsed Then a source entity is created and has findings | Validate the Grype parser works with a valid sample json file | Backend Engineer | Java Springboot Back-end  IntelliJ IDE or Jenkins Pipeline |
| **Test File:** src/test/java/com/zacharykarpinski/luminousonion/parser/GrypeTest.java | | | | |
|  | | | | |
|  | | | | |
| Integration Test | Given a new product json post message  When sent to the product api  Then create new product and send created status code | Validate the expected integration between the frontend sending a post create new product request to the backend using mocks. | Backend Engineer | Java Springboot Back-end  IntelliJ IDE or Jenkins Pipeline |
| **Test File:** src/test/java/com/zacharykarpinski/luminousonion/integration/ProductIntegrationTests.java | | | | |
|  | | | | |
|  | | | | |
| UI Unit Test Suite | Given a component test the logic and view are as expected. | Validate the UI components are displaying as expected. | UI Team | Test environment  React Front-end  Chrome Profiler Tool to measure response time |
| **Test files:** src/main/UI/src/components/VulnerabilityCard.test.js src/main/UI/src/components/Finding/FindingSummarySmall.test.js | | | | |
| **Results:** | | | | |

## 2. Full test run and result

Result from backend **.\mvn test**:



Result from frontend **yarn test**:



## 2. Changes made due to testing

After the extensive testing, I found the parsers were not working as expected. The structure of the json file was not being read properly, resulting in sources being created with one 1 finding, when there are more than 1. We identified the parser flaw and re-tested using our Grype unit test to validate the issue was remediated prior to pushes the code changes.

I also changed to a Test-Driven-Development approach where tests were written during or before development. That way, I built a test to match the requirement and the expected outcome. Then designed the class or function to satisfy the test.