**CS673 Software Engineering** 

**Team 4 - Rxcellent**

**Software Test Document**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Ryan Burns | Team Leader | *Ryan BUrns* | Dec 10, 2022 |
| Ignacio Joaquin Moral | Requirements Leader | *Ignacio Joaquin Moral* | Nov 7, 2022 |
| Zahit Odabas | QA Leader | *Zahit Odabas* | Dec 10, 2022 |
| Chenfei Yu | Configuration Leader | *Chenfei Yu* | Nov 9, 2022 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

[Testing Summary](#_sm5odwyvuk3j)

[Manuel Tests Reports](#_pqso2mbjyzx4)

[Automated Testing Reports](#_mtfbusfb0eq3)

[Testing Metrics](#_rijyjeu2ojqa)

[References](#_15tmymhipvdv)

[Glossary](#_8n34lvocupub)

# Testing Summary

**Vitest**

We mainly use Vitest.js library to implement and automate our test cases. We concentrated on backend testing for iteration 2. Backend developers (Zahit, Tsing, Ignacio) implemented the tests. We will implement frontend test cases for the next iteration, since team members have a little prior knowledge of React.js frontend testing.

We performed the tests that reside in 3 different categories. All these tests are performed before merging into the development branch.

Our test files reside in “src/test” directory.

**1- Unit Tests**

We tested our database object models in this category. Model fields are examined to prove their compliance with the defined requirements (type restrictions, uniqueness, necessity, etc..). For db manipulations, we used mongoose-in-memory-db library, which creates a temporary db for testing, to keep our main database clean.

**2- Integration Tests**

Our API tests fall into this category. We created sample requests referring to our API implementation and checked whether the request gets the expected response. We utilized Supertest.js library to mimic an end-to-end backend process. Our backend app takes the request, then it's routed to the router by path mapping, then the router forwards it to its related controller function. Finally the controller performs the db query with the intended db Model object and returns the response.

**3- Regression Tests**

This means re-running our existing test cases each time before committing to repo. Thus, we can see if a change in the code causes a test failure. If it does, we re-analyze our test case and decide whether we need to fix the code or change the test requirement.

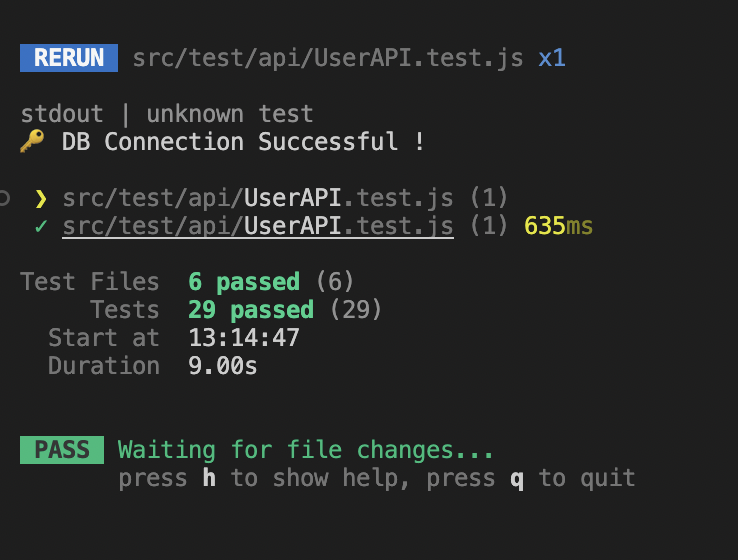
**4- Other QA related work**

We always create pull requests to merge our code into development branches. All members participated in the reviewing process but mainly QA leader (Zahit) and Design leader (Christol) performed code reviews.

If necessary, feedback and issues are created after code reviewing. Issues are assigned to the related team members in Github. We track our bugs and mark them as completed when they are resolved.

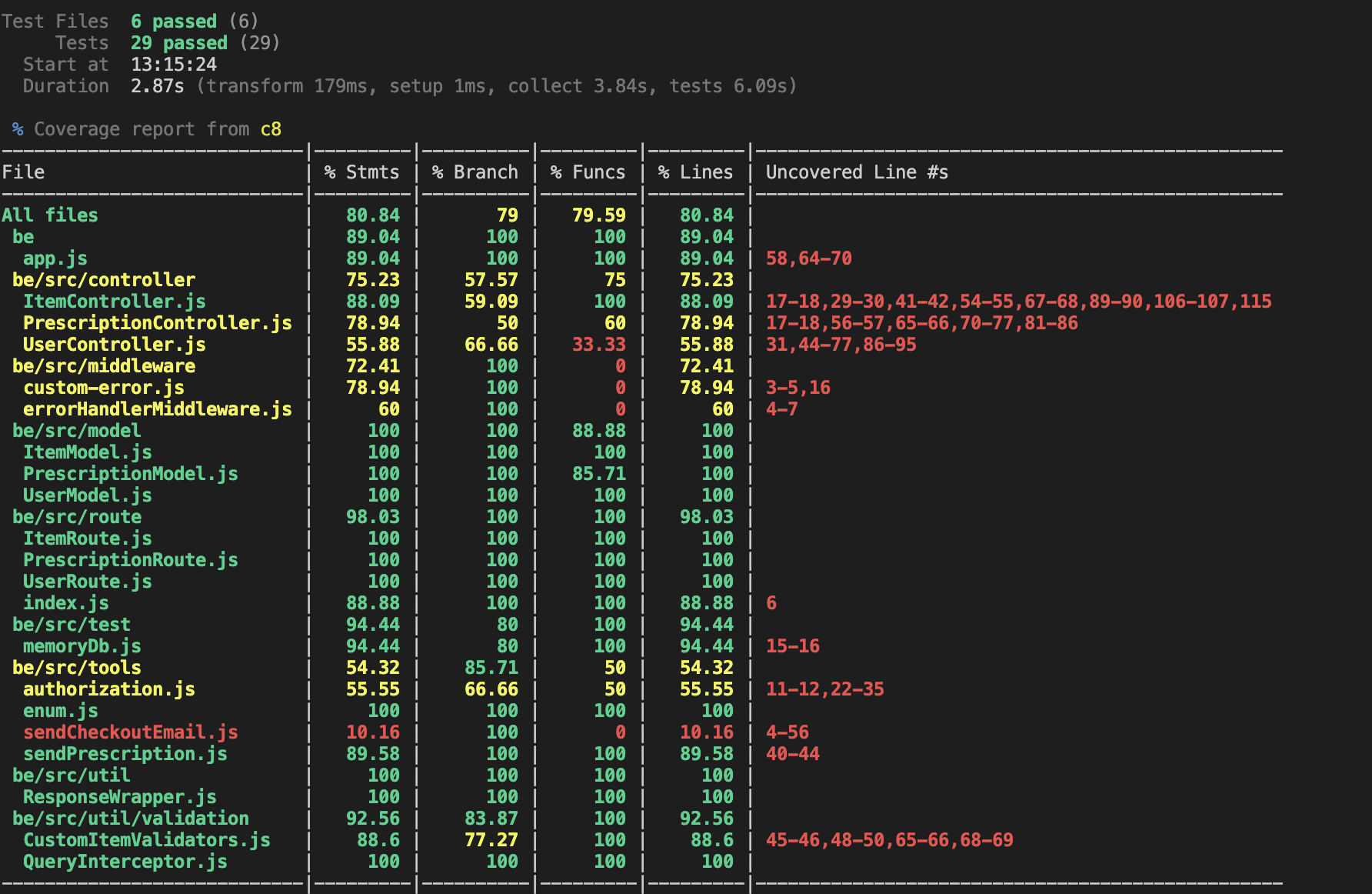
We performed refactoring to eliminate code smells and to follow our coding style. For example, we created a response format and applied it to all API responses to unify and standardize our response. Another one is, we created a request interceptor that utilizes Strategy Pattern to apply different validation functions for different types of requests.

# Automated Testing Report

****

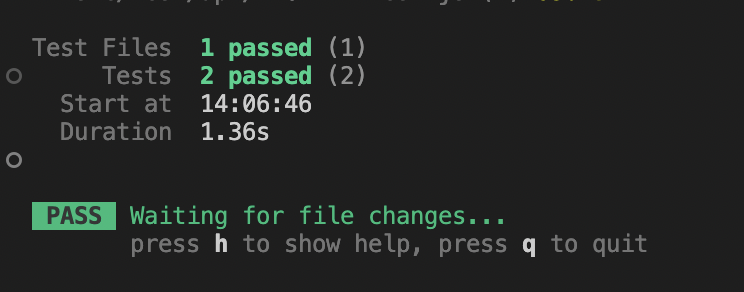
Above is the report from “npm run test” for backend.

We have 6 test files, a total of 29 test cases. 6 unit test cases and 19 integration test cases. All tests are passed.



Above is the result from “npm run coverage” for backend. We have more than 80% of total line coverage.

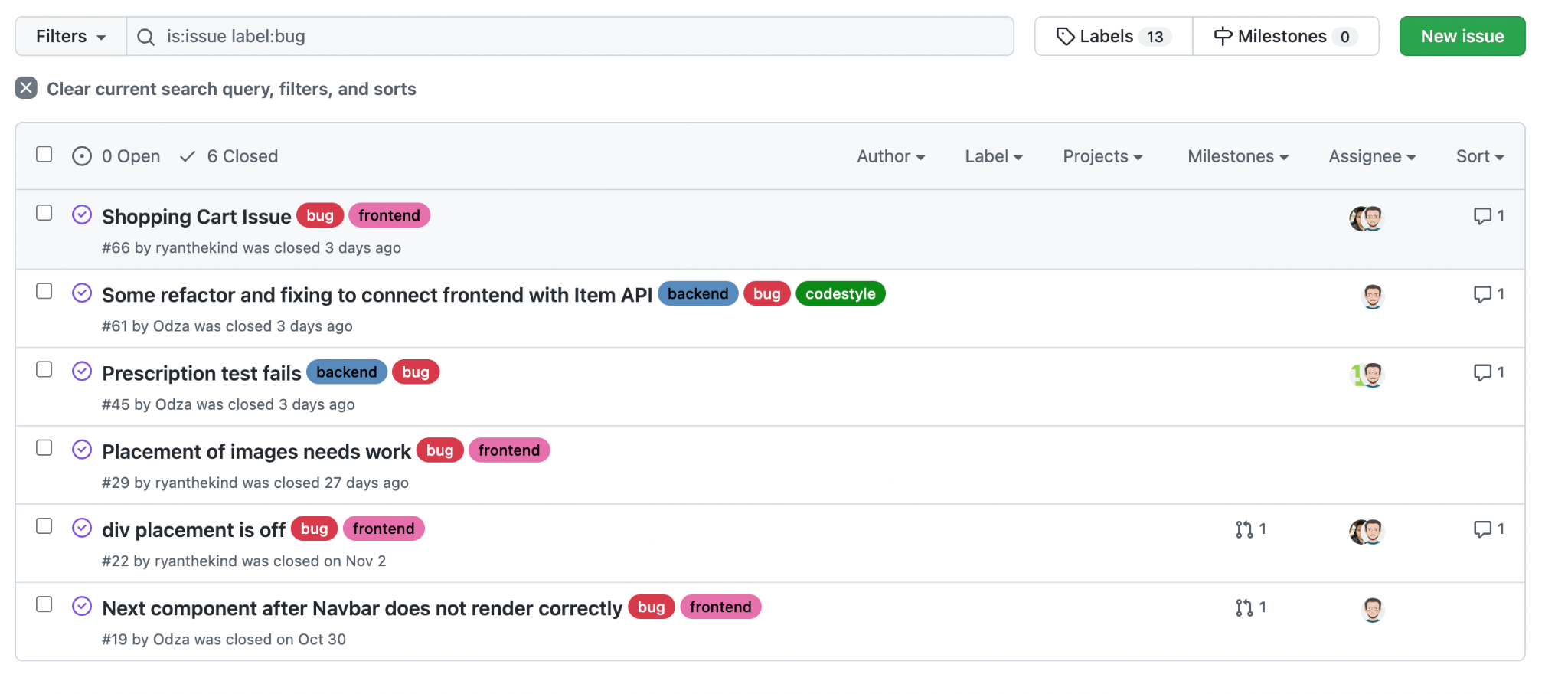
We have only 2 test cases for frontend. We haven’t got much time for frontend testing since we were mainly busy with feature implementations.



# Testing Metrics

| Metric Name | Description |
| --- | --- |
| # of test cases | 31 |
| test cases pass rate | 100% |
| # of resolved bugs | 6 of 6 tracked bugs |
| Test coverage rate | Backend -> 80.84% total line coverage  Backend -> 79.00% total branch coverage |
| # of critical bugs | <Not performed yet> |

We haven’t got a chance to perform critical bug testing.



# References

# Glossary