Unit Testing Report

December 1, 2024

Contents

Introduction

This document provides an overview of the unit testing conducted for the project using tools such as Jest, Vitest, and Istanbul. Covers back-end and front-end testing with details of coverage, methodology, and benefits.

1 Tools and Frameworks

Jest: A JavaScript testing framework designed to ensure the correctness of codebases.
Vitest: A fast, modern unit testing framework for front-end development.
Istanbul: Provides coverage statistics for JavaScript codebases.

2 White-Box Testing

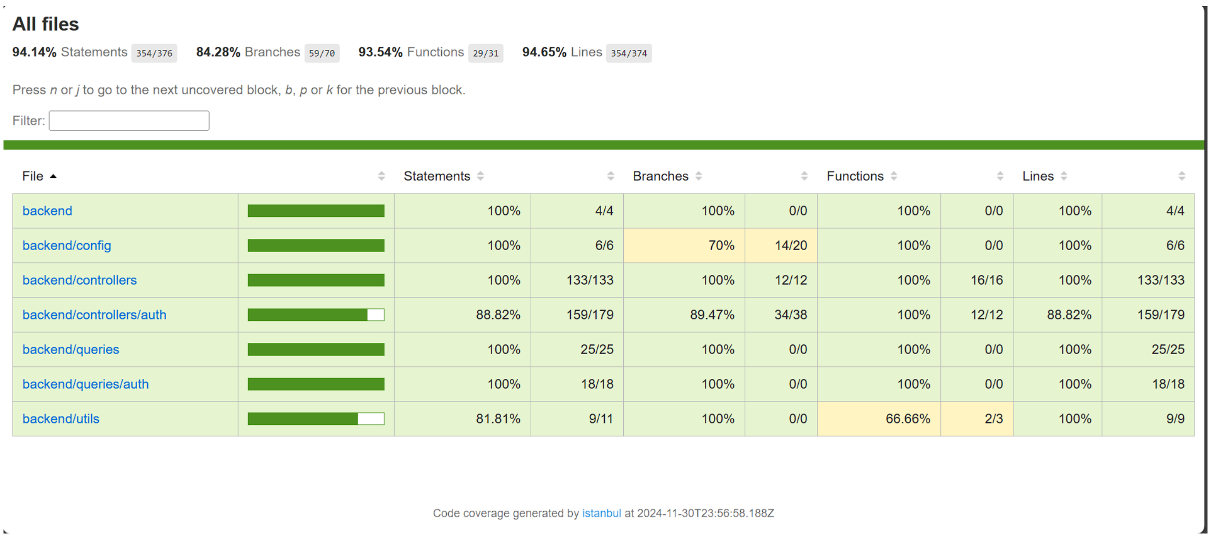
White-box testing involves analysing and verifying the internal structure and logic of the code. Unlike black-box testing, which focuses solely on input-output behavior, white-box testing examines the internal mechanics of the code.

3 Why is Unit Testing Important?

- **Early Bug Detection:** Identifies issues at an early stage, minimizing their impact on the project.
- **Enhanced Code Quality:** Encourages writing clean, modular, and maintainable code.
- **Faster Debugging:** Pinpoints issues when a test fails, speeding up the resolution process.

4 Backend Unit Testing using Jest

4.1 Overall Coverage



All files

94.14% Statements 354/376 **84.28%** Branches 59/70 **93.54%** Functions 29/31 **94.65%** Lines 354/374

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

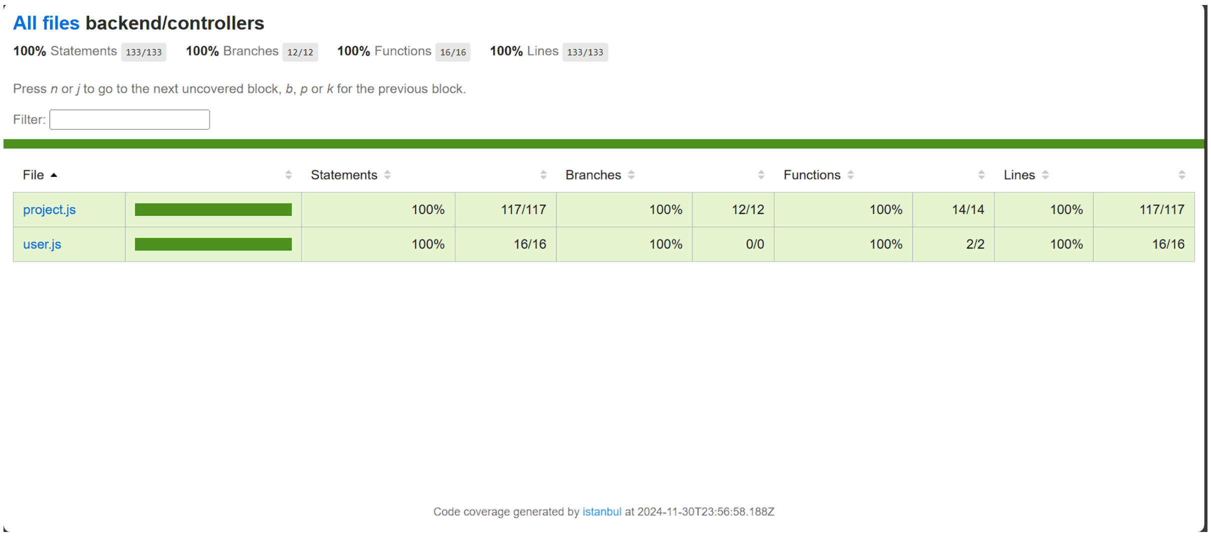
Filter:

File		Statements		Branches		Functions		Lines	
backend		100%	4/4	100%	0/0	100%	0/0	100%	4/4
backend/config		100%	6/6	70%	14/20	100%	0/0	100%	6/6
backend/controllers		100%	133/133	100%	12/12	100%	16/16	100%	133/133
backend/controllers/auth		88.82%	159/179	89.47%	34/38	100%	12/12	88.82%	159/179
backend/queries		100%	25/25	100%	0/0	100%	0/0	100%	25/25
backend/queries/auth		100%	18/18	100%	0/0	100%	0/0	100%	18/18
backend/utils		81.81%	9/11	100%	0/0	66.66%	2/3	100%	9/9

Code coverage generated by istanbul at 2024-11-30T23:56:58.188Z

Figure 1: Overall coverage for backend testing.

4.2 Coverage of Controllers



All files backend/controllers

100% Statements 133/133 **100%** Branches 12/12 **100%** Functions 16/16 **100%** Lines 133/133

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

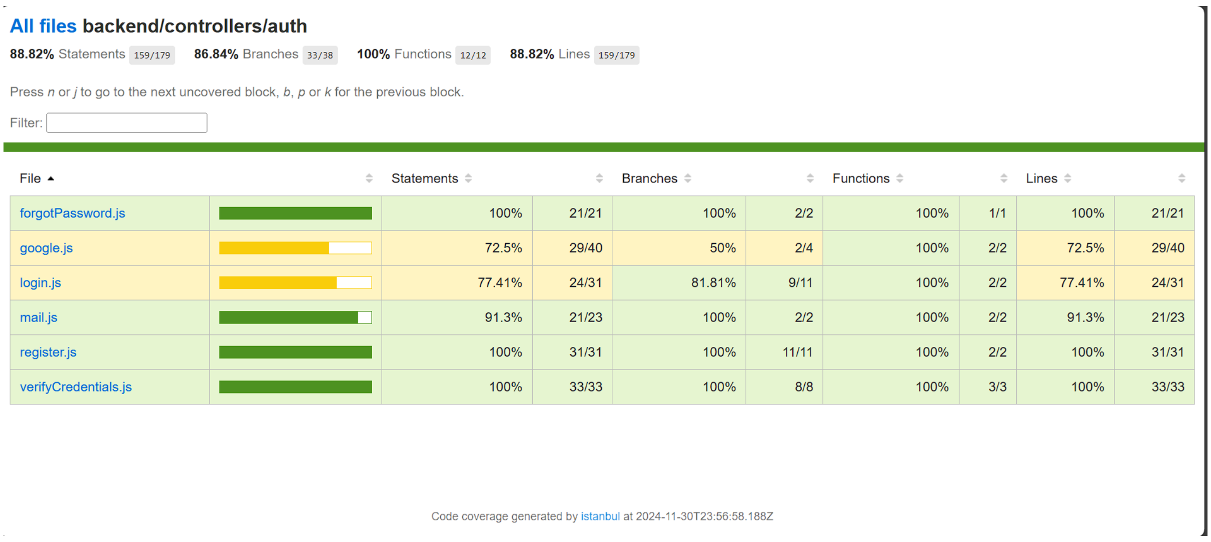
Filter:

File		Statements		Branches		Functions		Lines	
project.js		100%	117/117	100%	12/12	100%	14/14	100%	117/117
user.js		100%	16/16	100%	0/0	100%	2/2	100%	16/16

Code coverage generated by istanbul at 2024-11-30T23:56:58.188Z

Figure 2: Coverage of backend controllers.

4.3 Coverage of Authentication Controllers



All files backend/controllers/auth

88.82% Statements 159/179 **86.84%** Branches 33/38 **100%** Functions 12/12 **88.82%** Lines 159/179

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

Filter:

File		Statements		Branches		Functions		Lines	
forgotPassword.js		100%	21/21	100%	2/2	100%	1/1	100%	21/21
google.js		72.5%	29/40	50%	2/4	100%	2/2	72.5%	29/40
login.js		77.41%	24/31	81.81%	9/11	100%	2/2	77.41%	24/31
mail.js		91.3%	21/23	100%	2/2	100%	2/2	91.3%	21/23
register.js		100%	31/31	100%	11/11	100%	2/2	100%	31/31
verifyCredentials.js		100%	33/33	100%	8/8	100%	3/3	100%	33/33

Code coverage generated by istanbul at 2024-11-30T23:56:58.188Z

Figure 3: Coverage of authentication controllers.

5 Frontend Unit Testing Using Vitest

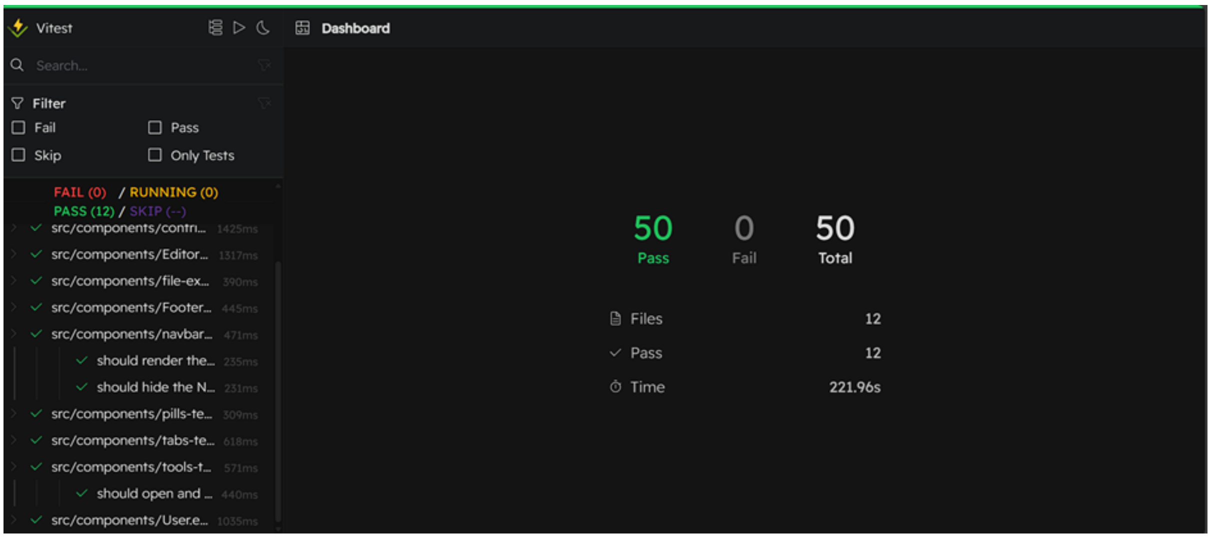
5.1 Why Use Vitest for Frontend Unit Testing?

Vitest is used for frontend unit testing to validate the functionality of individual React components. It offers:

- **Modern Syntax Support:** Compatible with ES modules, async / await, and other contemporary JavaScript features.
- **Built-in Coverage Reports:** Provides detailed coverage metrics to identify tested and untested code.
- **Fast Execution:** Runs tests in parallel, reducing the execution time.

5.2 Verified Test Cases

Vitest provides a user-friendly UI dashboard that allows users to monitor, analyse, and run tests effectively.



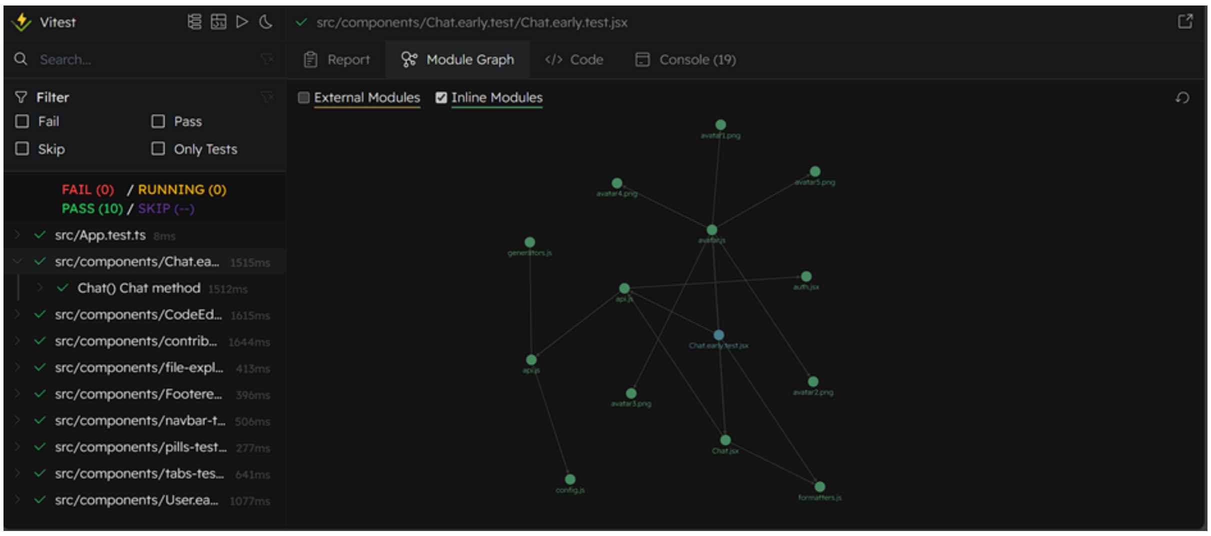
FAIL (0) / RUNNING (0)
PASS (12) / SKIP (--)

```
✓ src/components/contri...    1425ms
✓ src/components/Editor...    1317ms
✓ src/components/file-ex...    390ms
✓ src/components/Footer...     445ms
✓ src/components/navbar...     471ms
    ✓ should render the...     235ms
    ✓ should hide the N...     231ms
✓ src/components/pills-te...   309ms
✓ src/components/tabs-te...    618ms
✓ src/components/tools-t...    571ms
    ✓ should open and ...      440ms
✓ src/components/User.e...    1035ms
```

```
50      0      50
Pass    Fail   Total

Files          12
Pass           12
Time      221.96s
```

Figure 4: Vitest UI dashboard showing test results.



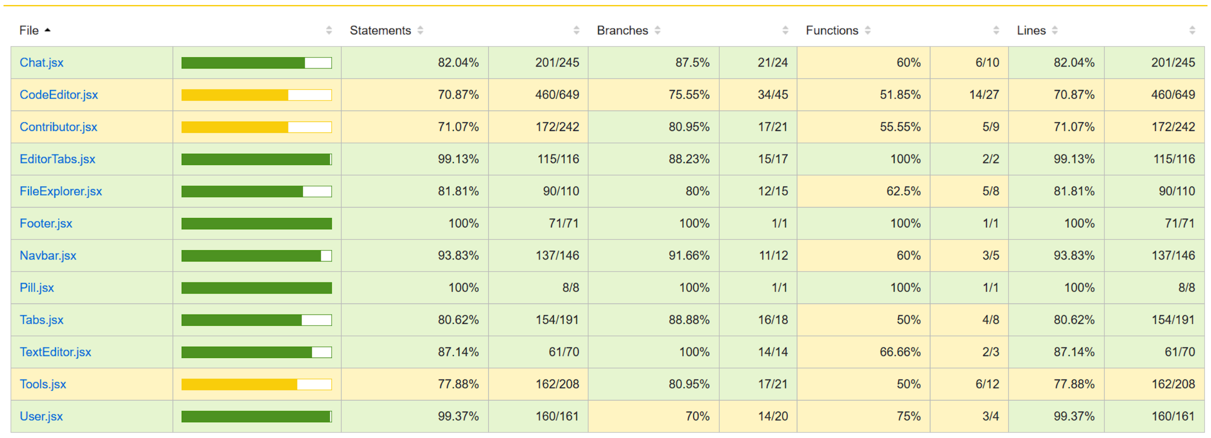
✓ src/components/Chat.early.test/Chat.early.test.jsx

FAIL (0) / RUNNING (0)
PASS (10) / SKIP (--)

```
✓ src/App.test.ts               8ms
✓ src/components/Chat.ea...  1515ms
    ✓ Chat() Chat method     1512ms
✓ src/components/CodeEd...   1615ms
✓ src/components/contrib...  1644ms
✓ src/components/file-expl... 413ms
✓ src/components/Footere...   396ms
✓ src/components/navbar-t...  506ms
✓ src/components/pills-test... 277ms
✓ src/components/tabs-tes...  641ms
✓ src/components/User.ea...  1077ms
```

Figure 5: Dependency module graph of test files.

5.3 Coverage of React Components



File		Statements		Branches		Functions		Lines	
Chat.jsx		82.04%	201/245	87.5%	21/24	60%	6/10	82.04%	201/245
CodeEditor.jsx		70.87%	460/649	75.55%	34/45	51.85%	14/27	70.87%	460/649
Contributor.jsx		71.07%	172/242	80.95%	17/21	55.55%	5/9	71.07%	172/242
EditorTabs.jsx		99.13%	115/116	88.23%	15/17	100%	2/2	99.13%	115/116
FileExplorer.jsx		81.81%	90/110	80%	12/15	62.5%	5/8	81.81%	90/110
Footer.jsx		100%	71/71	100%	1/1	100%	1/1	100%	71/71
Navbar.jsx		93.83%	137/146	91.66%	11/12	60%	3/5	93.83%	137/146
Pill.jsx		100%	8/8	100%	1/1	100%	1/1	100%	8/8
Tabs.jsx		80.62%	154/191	88.88%	16/18	50%	4/8	80.62%	154/191
TextEditor.jsx		87.14%	61/70	100%	14/14	66.66%	2/3	87.14%	61/70
Tools.jsx		77.88%	162/208	80.95%	17/21	50%	6/12	77.88%	162/208
User.jsx		99.37%	160/161	70%	14/20	75%	3/4	99.37%	160/161

Figure 6: Coverage of React components.

Conclusion

Unit testing is crucial to ensure the quality, maintainability, and reliability of the project. Using Jest, Vitest, and Istanbul, comprehensive testing was performed to effectively cover back-end and front-end functionalities.