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

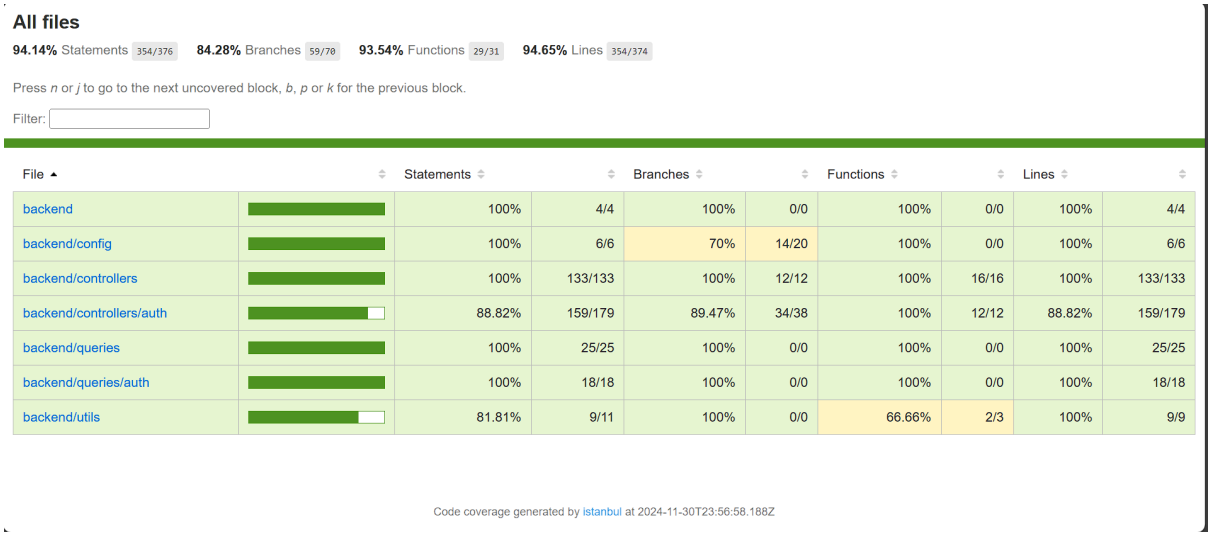


Figure 1: Overall coverage for backend testing.

4.2 Coverage of Controllers

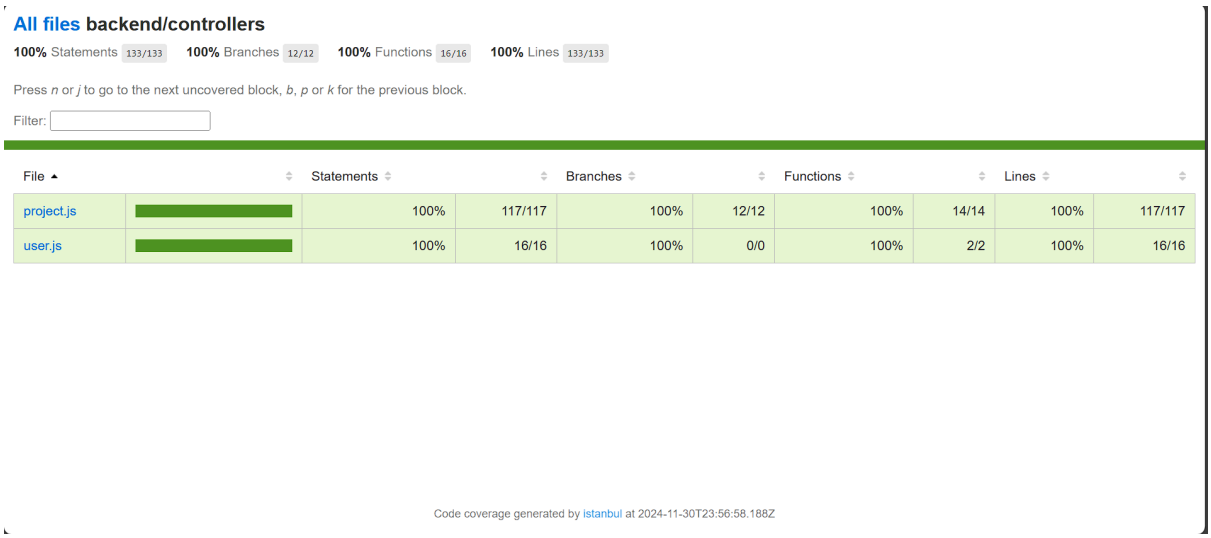


Figure 2: Coverage of backend controllers.

4.3 Coverage of Authentication Controllers

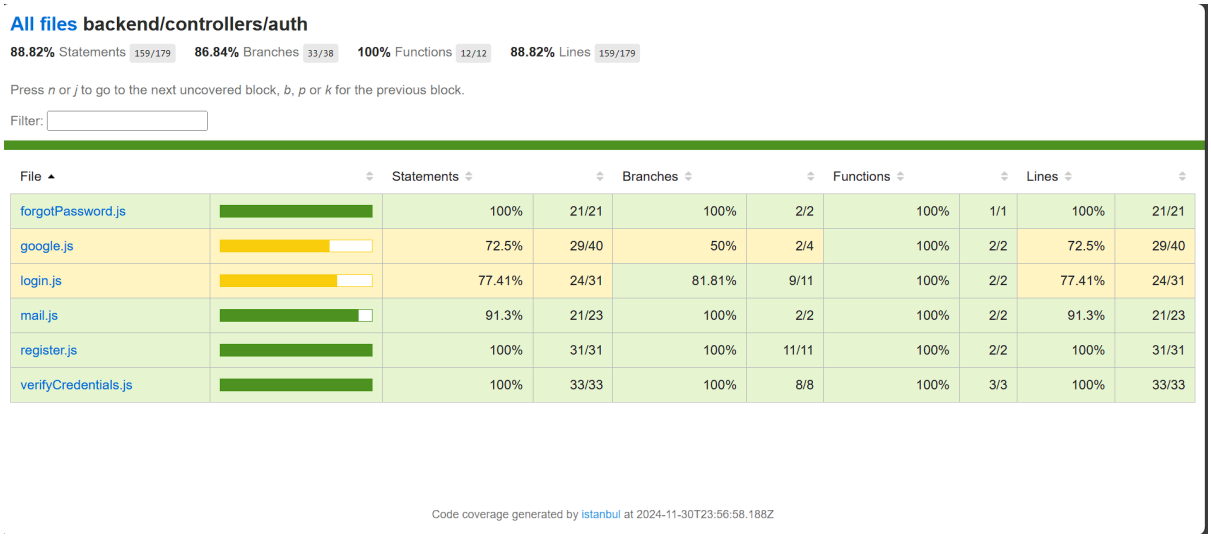


Figure 3: Coverage of authentication controllers.

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.

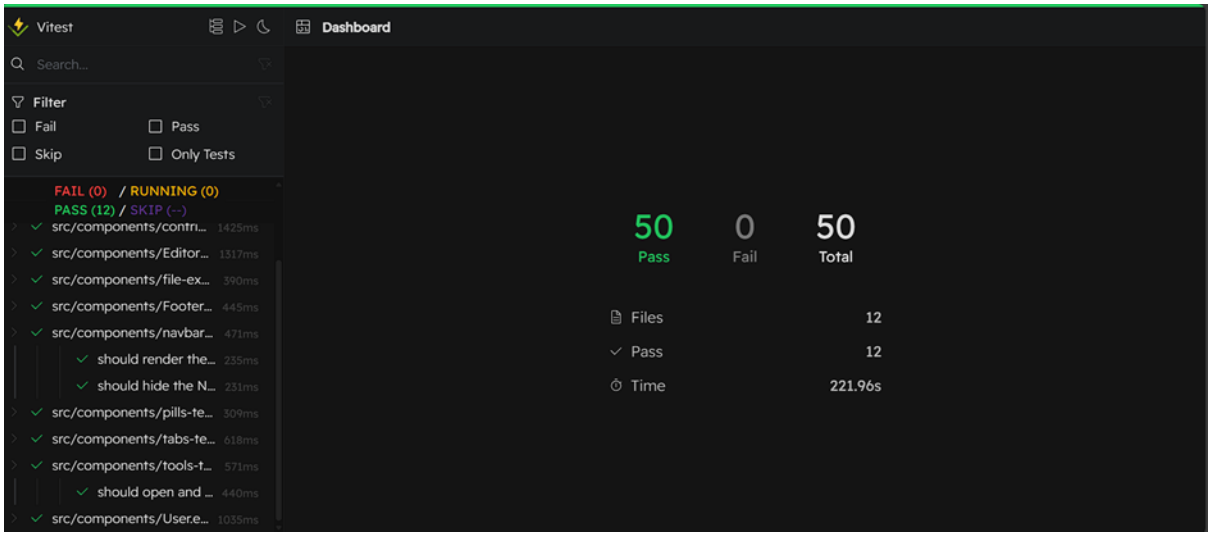


Figure 4: Vitest UI dashboard showing test results.

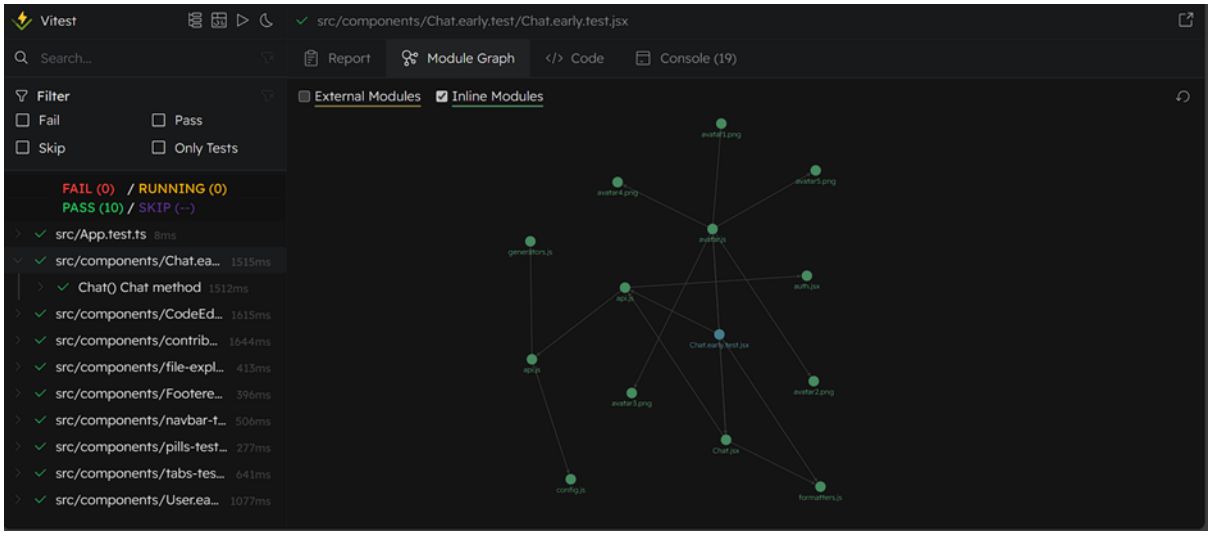
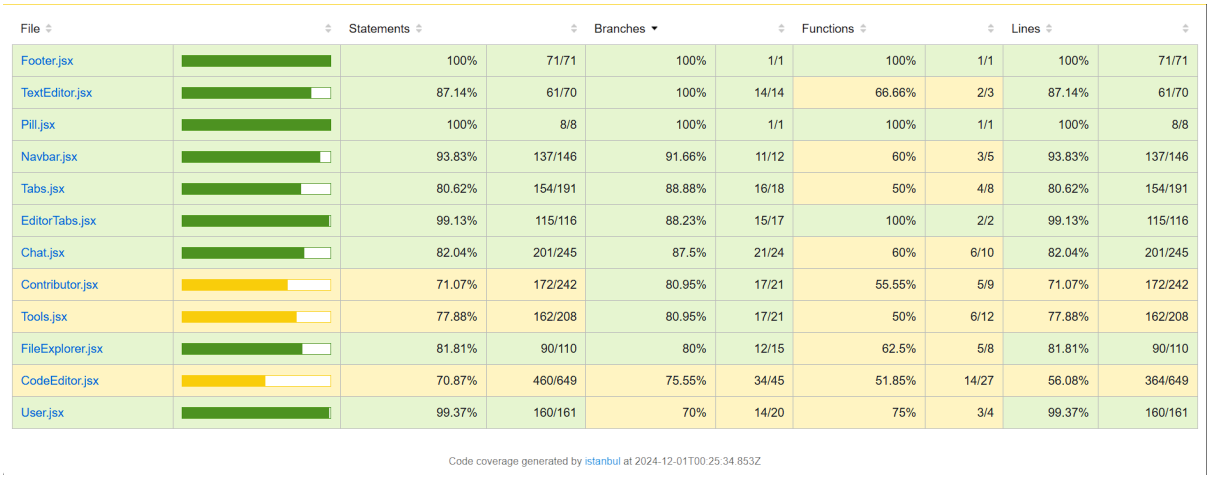


Figure 5: Dependency module graph of test files.

5.3 Coverage of React Components



File	Statements		Branches		Functions		Lines	
Footer.jsx	100%	71/71	100%	1/1	100%	1/1	100%	71/71
TextEditor.jsx	87.14%	61/70	100%	14/14	66.66%	2/3	87.14%	61/70
Pill.jsx	100%	8/8	100%	1/1	100%	1/1	100%	8/8
Navbar.jsx	93.83%	137/146	91.66%	11/12	60%	3/5	93.83%	137/146
Tabs.jsx	80.62%	154/191	88.88%	16/18	50%	4/8	80.62%	154/191
EditorTabs.jsx	99.13%	115/116	88.23%	15/17	100%	2/2	99.13%	115/116
Chat.jsx	82.04%	201/245	87.5%	21/24	60%	6/10	82.04%	201/245
Contributor.jsx	71.07%	172/242	80.95%	17/21	55.55%	5/9	71.07%	172/242
Tools.jsx	77.88%	162/208	80.95%	17/21	50%	6/12	77.88%	162/208
FileExplorer.jsx	81.81%	90/110	80%	12/15	62.5%	5/8	81.81%	90/110
CodeEditor.jsx	70.87%	460/649	75.55%	34/45	51.85%	14/27	56.08%	364/649
User.jsx	99.37%	160/161	70%	14/20	75%	3/4	99.37%	160/161

Code coverage generated by istanbul at 2024-12-01T00:25:34.853Z

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.