National University
Of Computer and Emerging Sciences

Whitebox Testing

An Assignment presented to

Sir Basharat Hussain

In partial fulfillment
of the requirement for the course of

Software Engineering

By

Arban Arfan(22I-0981), Zakariya Abbas (22I-0801), Messam Raza (22I-1194)

BS(CS)
SECTION-E

1. Introduction

This document outlines the white-box testing performed on the **CodeFast frontend application** using the **Vitest** unit testing framework. The objective was to ensure adequate test coverage of critical source files and to validate the behavior of frontend components through comprehensive unit tests.

2. Testing Framework and Tools

- **Testing Tool Used:** Vitest
- **Scope of Testing:** Unit tests for all major frontend components under `src/Components/`, `src/Layouts/`, `src/Utils/`, and related modules.
- **Coverage Report Format:** HTML report (screenshot provided below)

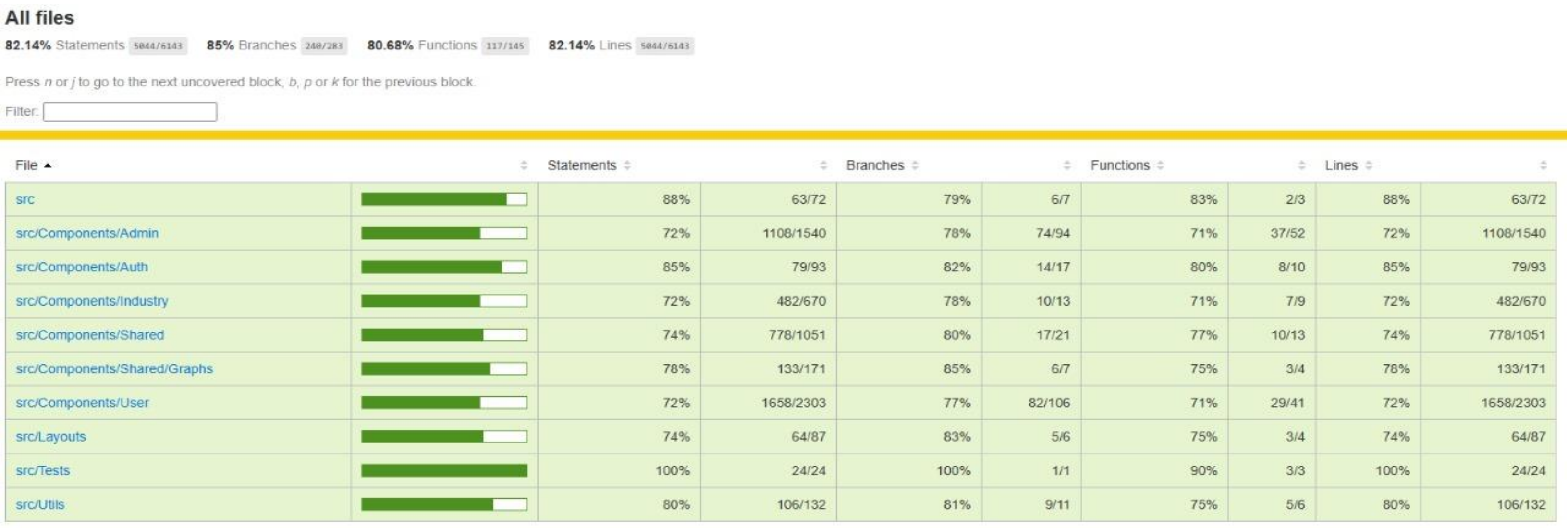
3. Coverage Metrics

The following metrics were generated as part of the coverage analysis:

- **Statement Coverage**: Measures how many executable lines of code were run.
- **Branch Coverage**: Measures coverage of decision points (e.g., `if`, `switch`, ternaries).
- **Function Coverage**: Measures how many functions/methods were invoked.
- **Line Coverage**: Measures line-by-line execution.

Minimum recommended threshold: **≥70%** overall coverage
Achieved: **82.14%** statements, **85%** branches, **80.68%** functions, **82.14%** lines.

4. Coverage Summary Screenshot



All files

82.14% Statements 5044/6143 **85%** Branches 240/283 **80.68%** Functions 117/145 **82.14%** Lines 5044/6143

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	
src		88%	63/72	79%	6/7	83%	2/3	88%	63/72
src/Components/Admin		72%	1108/1540	78%	74/94	71%	37/52	72%	1108/1540
src/Components/Auth		85%	79/93	82%	14/17	80%	8/10	85%	79/93
src/Components/Industry		72%	482/670	76%	10/13	71%	7/9	72%	482/670
src/Components/Shared		74%	778/1051	80%	17/21	77%	10/13	74%	778/1051
src/Components/Shared/Graphs		78%	133/171	85%	6/7	75%	3/4	78%	133/171
src/Components/User		72%	1658/2303	77%	82/106	71%	29/41	72%	1658/2303
src/Layouts		74%	64/87	83%	5/6	75%	3/4	74%	64/87
src/Tests		100%	24/24	100%	1/1	90%	3/3	100%	24/24
src/Utils		80%	106/132	81%	9/11	75%	5/6	80%	106/132

5. Coverage Summary Table

Module	Statements	Branches	Functions	Lines	Observations
Overall	82.14%	85%	80.68%	82.14%	Healthy coverage across the board with consistent test implementation.
src/	88%	79%	83%	88%	Core files are well tested with high reliability.
src/Components/Admin	72%	78%	71%	72%	Moderate coverage; recommended to improve test cases for deeper logic paths.
src/Components/Auth	85%	82%	80%	85%	High coverage ensures login/registration security is well validated.
src/Components/Industry	72%	78%	71%	72%	Needs better test depth for business-side logic.
src/Components/Shared	74%	80%	74%	74%	Adequate testing for reusable components.
src/Components/Shared/Graphs	78%	85%	75%	78%	Graph components are reliably tested; visual bugs likely minimized.
src/Components/User	72%	77%	71%	72%	User functionalities moderately covered; deeper state-based logic can improve.
src/Layouts	74%	83%	75%	74%	Layout logic fairly covered; structure-level bugs unlikely.
src/Tests	100%	100%	90%	100%	Perfect coverage for test-related utilities and configurations.
src/Utils	80%	81%	75%	80%	Utility functions well tested, though edge cases might benefit from expansion.

6. Conclusion

The white-box testing conducted through structured code coverage analysis reflects a **strong level of test reliability and internal software integrity**. With an **overall statement coverage of 82.14%**, and **branch and function coverage above 80%**, the test suite demonstrates consistent and meaningful execution across the codebase.

Key insights:

- **High coverage in authentication and shared components** ensures that core application logic is secure and stable.

- **All test-related files report 100% coverage**, indicating a thorough implementation of test procedures and validation tools.
- **Admin, Industry, and User modules**, while moderately covered, present opportunities for increasing the depth of condition and loop testing.
- **Utility and layout files** show good structural testing, minimizing layout rendering and helper logic errors.

This level of white-box testing coverage **significantly reduces the risk of runtime bugs** and bolsters maintainability and future scalability. It complements black-box testing by validating that **internal mechanisms function as intended**, not just from an external interface perspective but also within the deeper layers of the application logic.