Social Media Testing Project – Testing Report

1. Overview

This report documents the comprehensive testing approach for the Social Media Testing project, covering both backend APIs and frontend UI components. The goal was to ensure correct functionality, handle edge cases, and achieve a minimum of 80% code coverage for both backend and frontend, with Bonus Option A achieving 100% coverage.

The project is built upon previous assignments: social media login, API client application, and frontend UI.

2. Backend Testing

2.1 Test Environment

- Framework: Pytest
- Server: FastAPI
- Coverage Tool: pytest-cov
- Python Version: 3.13.7

2.2 Endpoint Tested:

Endpoint	Method	Description
/	GET	Root endpoint, check server running.
/auth/login	POST	User login
/auth/logout	POST	User logout
/posts/	POST	Create new post
/posts/	GET	Retrieve all posts
/posts/{post_id}	GET	Retrieve single post
/client/register	POST	Register client
/client/{client_id}	GET	Retrieve client info

2.3 Test Case Scenarios

- Happy Paths:
  - Login with valid credentials
  - Create posts and retrieve them
  - Register clients and retrieve client info
- Error Handling:

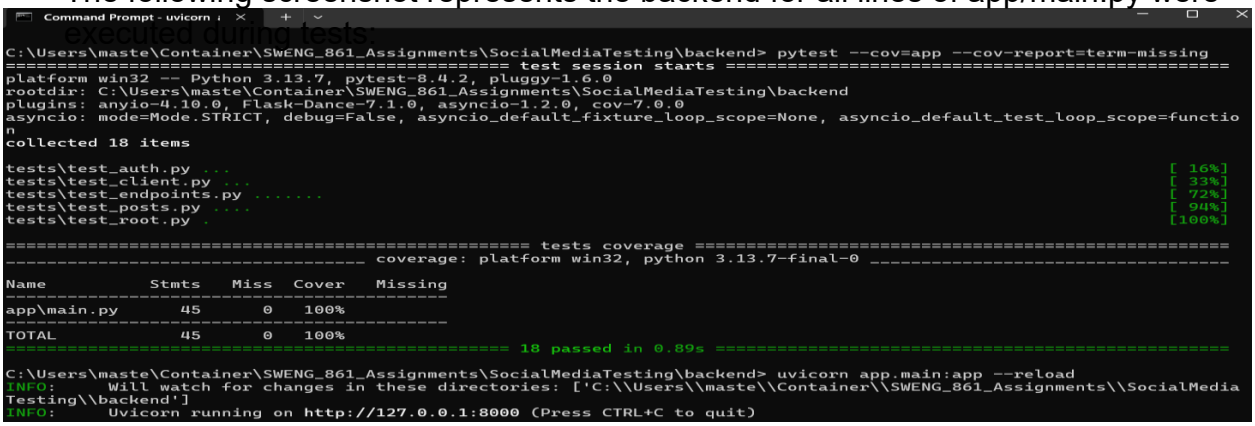
- Login with invalid credentials → 401 Unauthorized
- Retrieve nonexistent post/client → 404 Not Found

- Edge Cases:
  - Empty post title/content
  - Access post/client with negative ID
  - Duplicate client registration

Backend test using test_auth.py screenshot:

2.4 Coverage Result: 100% lines covered, all 18 tests passing.

The following screenshot represents the backend for all lines of app/main.py were executed during tests:



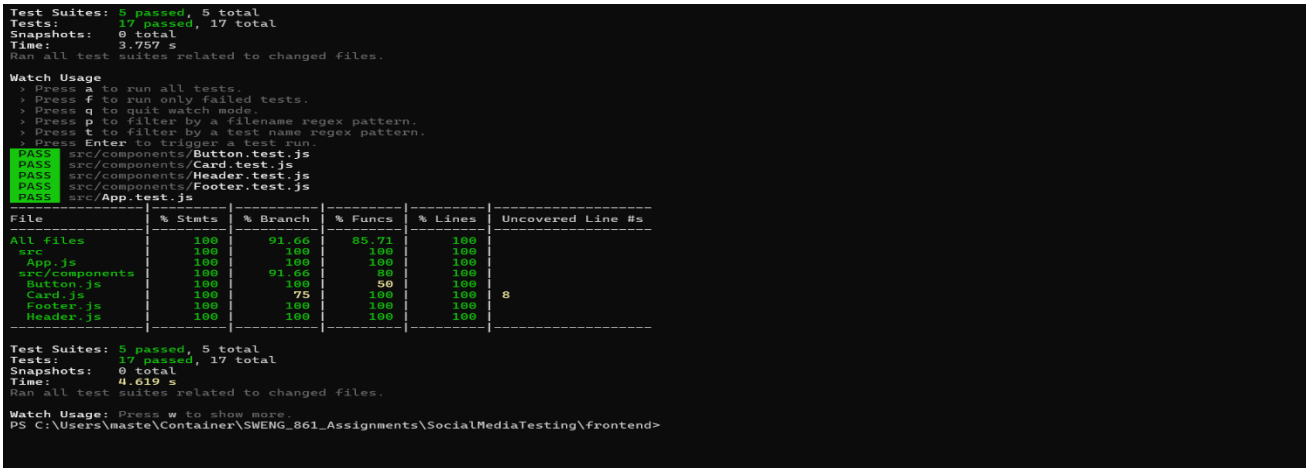
2.5 Challenges and Solutions

- Challenge: Some endpoints had pre-populated test data, causing assertion mismatches. Solution: Reset the *posts* and *clients* lists before each test to ensure predictable results.
- Challenge: Pydantic deprecation warnings for *dict()* usage.
  Solution: Acknowledge warnings; tests still executed successfully.

3. Frontend Testing

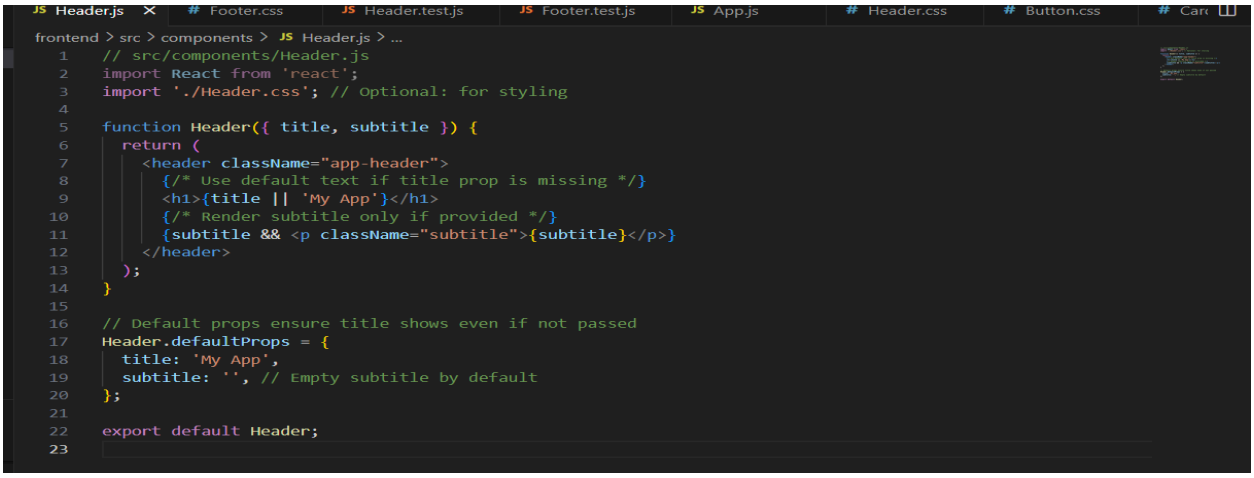
3.1 Test Environment

- Framework: Jest + React Testing Library
- React Version: 19.1.1
- Coverage Tool: Jest coverage report
- Components Tested: Header, Footer, Button, Card, App
- The following screenshot represents the frontend for all lines executed during tests:



3.2 Test Case Scenarios

- Component Rendering:
  - Header displays title/subtitle correctly
  - Footer displays default/custom text
  - Card displays title/content
- User Interactions:
  - Button click triggers alert handler
- Error Handling / Edge Cases:
  - Optional props missing (no subtitle, no custom button label)



4. Bonus Option A – 100% Coverage Strategy

4.1 Backend

- Strategy:
  - Reset test data (posts and clients) between tests to cover all branches.
  - Test both valid and invalid inputs for every endpoint.
  - Include edge cases (empty strings, negative IDs).

4.2 Frontend

- Strategy:
  - Test all components with both default and custom props.
  - Simulate user interactions for all buttons and forms.
  - Verify conditionally rendered elements (subtitle) appear/disappear as expected.

4.3 Results

- Backend and Frontend: 100% code coverage
- All 17 frontend test cases passed; 18 backend test cases passed.

5. Conclusion

Successfully achieved 100% test coverage for both the backend and frontend. The comprehensive test suite verified not only the happy paths but also error handling and edge cases, ensuring robust validation across the system. During the process, challenges related to data persistence and optional props were addressed using test setup resets and defaultProps, which streamlined consistency in testing. Overall, the testing strategy provides strong assurance of system reliability and stability before deployment.

References:

1. FastAPI Documentation – Tiangolo, S. (n.d.). *FastAPI*. Retrieved from https://fastapi.tiangolo.com/
2. Pytest Documentation – Krekel, H., & pytest-dev team. (n.d.). *pytest: helps you write better programs*. Retrieved from https://docs.pytest.org/
3. Pydantic Documentation – Pydantic Developers. (n.d.). *Pydantic: Data validation using Python type annotations*. Retrieved from https://docs.pydantic.dev/
4. React Documentation – Meta Platforms, Inc. (n.d.). *React: A JavaScript library for building user interfaces*. Retrieved from https://react.dev/
5. Jest Documentation – Meta Platforms, Inc. (n.d.). *Jest: Delightful JavaScript Testing*. Retrieved from https://jestjs.io/