Comprehensive documentation explaining

how MSW, Selenium, Mocha patterns, and other testing

tools relate in the Epic Stack context. Here's what

we have:

📚 Documentation Created:

1. TESTING-ARCHITECTURE.md

- Complete testing pyramid and architecture overview

- Tool relationships and integration points

- Test execution flows

- Comparison of testing tools (removed Playwright

references)

- Focus on Selenium as the primary E2E tool

2. MSW-DEEP-DIVE.md

- Detailed explanation of MSW (Mock Service Worker)

- Architecture diagrams showing browser and Node.js

integration

- Request/response flow diagrams

- Handler types and patterns

- Integration with Epic Stack commands

3. SELENIUM-MSW-INTEGRATION.md

- How Selenium and MSW work together

- Benefits of combining both tools

- Common testing patterns

- Best practices and debugging guides

🔑 Key Relationships Explained:

MSW (Mock Service Worker)

- Intercepts network requests at the service worker

level

- Provides consistent mock data for both development

and testing

- Works with both Selenium tests and Vitest unit

tests

- Controlled via MOCKS environment variable

Selenium WebDriver

- Primary E2E testing tool for browser automation

- Uses explicit wait strategies

- Can work with or without MSW mocks

- Organized using Mocha-style patterns (without the

Mocha dependency)

Mocha Patterns

- Not actually using Mocha library

- Pattern copied for test organization (describe/it

blocks)

- Implemented manually in Selenium exercises

- Provides familiar structure for test suites

Vitest

- Unit and integration testing

- Works with MSW for API mocking

- Fast execution with ESM support

- Tests colocated with source files

🎯 In Epic Stack Context:

1. Development: npm run dev starts the app with MSW

mocks enabled

2. Testing: Selenium tests can run against mocked or

real APIs

3. Mocking: MSW provides network-level mocking for

consistency

4. Organization: Tests use Mocha-style patterns for

clarity

The documentation includes numerous Mermaid diagrams

illustrating data flows, architecture, and

relationships between all testing tools in the Epic

Stack ecosystem.