**then() inside AuthService not use**

our manager’s recommendation — to keep then() inside AuthService — **goes against common testing and software design best practices**, **but it might be justified in certain tightly scoped, test-only frameworks**. Let's break it down honestly and objectively:

**🔍 Manager's Reasoning:**

*"It offers better modularity, maintainability, test clarity, and industry alignment."*

**✅ Possible Valid Contexts:**

* The AuthService is part of a **test-only utility layer**, **not used in production**.
* The service methods return void or perform internal assertions to **simplify tests**.
* There’s a **consistent pattern** across the team/framework for asserting inside helper methods.
* They want to **reduce repetition** in test cases (DRY principle).

In such controlled test frameworks, combining action + assertion may lead to **cleaner test methods**, especially for **high-volume API testing**.

**❌ Why It’s Generally a Bad Practice (In Broader Industry Context):**

1. **Violates Separation of Concerns**:
   * AuthService should focus on "doing" (e.g., making login calls), not "verifying" results.
   * Mixing behavior (login) with assertions (status check) reduces reusability and clarity.
2. **Limits Flexibility**:
   * What if some tests want to validate statusCode 200, and others expect 401?
   * Hard-coding then().statusCode(200) in AuthService forces test logic into infrastructure.
3. **Harder to Debug & Extend**:
   * Test logic spread across service methods hides the actual test intent.
   * Future custom validation (e.g., checking token format or response body) becomes harder.
4. **Industry Norm**:
   * In **most mature testing frameworks** (JUnit, TestNG, Rest-Assured, Mockito), the **assertions live in the test classes**, not in helper services.
   * Frameworks like Spring Boot, Rest-Assured, and Selenium follow this separation pattern.

**🟡 Final Evaluation:**

| **Question** | **Answer** |
| --- | --- |
| **Is your manager’s approach “wrong”?** | ❌ Not necessarily, if the context is test-only and tightly controlled. |
| **Is it aligned with broader industry best practices?** | ❌ No, it’s a deviation from typical modular design principles. |
| **Is it maintainable long-term?** | ⚠️ Maybe, but it introduces rigidity. |
| **Is your approach (assert in test class) better long-term?** | ✅ Yes, especially if test needs evolve or become more complex. |

**✅ Recommendation:**

You should respectfully clarify:

* *If AuthService is strictly for test automation, you understand and will follow the team's pattern.*
* But suggest keeping logic flexible by perhaps **overloading** methods—one version with then(), one returning Response.

Would you like help drafting a diplomatic message to raise this with your manager?