vention

🧪 Task 8: Data Validation Library

In this task, we'll use Cursor IDE to build and test a reusable data validation library for complex inputs.

📚 Theory

In this challenge, you'll use Cursor IDE not just as a coding tool, but as an intelligent assistant to help you design and implement a reusable validation library from scratch.

🧠 AI Technique

This task explores how AI-powered development can assist in three key areas:

🔧 AI-Assisted Validator Design

Cursor can help you build type-safe validator functions for primitive types (like string, number, boolean) and guide you through creating validators for more complex structures (arrays, nested objects, optional fields, etc.). You'll learn to prompt the AI to design reusable patterns that align with your chosen language's type system and conventions.

🛠️ AI-Driven Refactoring and Documentation

As your library grows in complexity, maintaining clean and well-documented code becomes essential. Use Cursor to generate inline documentation, usage examples, and even refactor parts of your codebase for better readability, modularity, and adherence to best practices.

☑️ Smart Test Generation

Validating the validator is just as important. You'll use AI to help create a comprehensive test suite with examples of valid and invalid data inputs. These tests will serve both as correctness checks and usage documentation for future developers.

📝 Task

Your task is to build a robust validation library in your preferred programming language (JavaScript, Python, Java, etc.) that can validate complex data structures leveraging Cursor IDE's capabilities.

Start with this basic template that you can adapt to your language of choice.

🎯 Implement Core Validator Functions

- Use Cursor's AI to help you write type-safe validator functions for primitive types (e.g., string, number, boolean).
- Extend your library to support complex types such as arrays and objects.
- Ensure your validators follow your language's best practices and type system.

📄 Leverage AI for Documentation and Refactoring

- Use Cursor's AI to generate inline documentation and usage examples.
- Refactor your code with AI suggestions to ensure clarity, maintainability, and adherence to your style guide.

🧪 Test with Diverse Data Patterns

- Write comprehensive test cases covering valid and invalid data scenarios.
- Use Cursor's AI to generate and expand your test suite, ensuring robust validation coverage.

📌 Requirements

- Validation library implementation should have complete, type-safe validator functions for primitive and complex types
- Validation library code should have inline comments explaining what this or that function does
- Guide (readme.md) should contain clear instructions on how to run the application and use your validator library
- Unit tests should cover all core functionality, and test coverage should be at least 60%