MPEG Sorter Development Prompt

I need a Python utility script that sorts media files based on their actual file signatures rather than relying on file extensions, with the following requirements:

Core Functionality

- Analyze files by signature (magic bytes) to identify MP3 and MP4 files regardless of extension
- Move identified MP3 files to an "audio" subdirectory and MP4 files to a "video" subdirectory
- Rename files with incorrect extensions (MP3 files saved as .mp4 and vice versa)
- Handle unknown file types by moving them to an "unknown" subdirectory (enabled by default)
- Create destination subdirectories within the source folder, not elsewhere
- Use `shutil.move()` to move files, not just copy them

Performance Requirements

- Implement both parallel and sequential processing modes for comparison
- Use asynchronous parallel processing with ThreadPoolExecutor for improved performance
- Make worker count configurable, defaulting to CPU core count
- Include progress tracking for large file collections
- Report performance metrics (files/second, total time)

Technical Requirements

- All lines should be within 120 characters
- Follow PEP8 import rules and formatting
- Include proper error handling for all file operations
- Validate input parameters and handle edge cases
- Provide clear, informative output on processing operations
- Handle filename conflicts when moving files

Command-line Interface

- Required argument: folder path for processing
- Optional argument: `--no-unknown` to disable unknown folder (default is enabled)
- Optional argument: `--workers` to set maximum number of concurrent workers
- Optional argument: `--sequential` to run in single-threaded mode for benchmarking

Testing Framework

- Create a single, comprehensive pytest-compatible test file (test_mpeg_sorter.py)
- Include automatic test data generation with appropriate file signatures
- Test both sequential and parallel processing modes
- Validate file sorting and extension correction
- Clean up after tests to restore original file structure
- Include performance comparison between modes
- Implement tests that work with both function calls and command-line invocation
- Ensure tests can be run directly or via pytest without modification

Documentation

- Create two separate documentation files:

README.md with a logical information flow:

1. Title and copyright notice (Eric Gitonga, MIT License)
2. Description of what MPEG Sorter is and does
3. Features list (clearly list parallel processing as an implemented feature)
4. Installation instructions
5. Usage information with options and examples
6. Output description
7. Testing information (place at the end)

Technical Documentation.md with a logical information flow:

1. Main MPEG Sorter overview with copyright notice
2. Implementation details (file structure, dependencies)
3. Core components and functionality
4. Signature detection details
5. Analysis of strengths and weaknesses (ensure parallel processing is listed as a strength, not a weakness)
6. Future improvements (remove any mention of parallel processing as it's already implemented)
7. Implementation considerations
8. Testing framework and details (place at the end)

Project Structure

```
mpeg-sorter/              - Module directory
  mpeg_sorter.py          - Main Python script
tests/                    - Test directory
  conftest.py             - Pytest configuration
  test_mpeg_sorter.py     - Pytest-compatible test script
  data/                   - Test data directory
```

Please develop the script, test suite, and documentation as specified above. Ensure consistency between the documentation and the actual implementation, particularly regarding the parallel processing feature which should be clearly identified as an implemented strength, not a future improvement.