Validata

.NET Task: CSV Reconciliation Tool (multithreaded)

Goal:
Build a .NET console/library tool that reconciles CSV data between two folders (each containing 5 CSV files). The tool must compare files and produce a reconciliation report showing matched and unmatched records. It must use multithreading to speed up processing.

Requirements (functional)

1. **Input**
    - Two folder paths (FolderA, FolderB). Each folder contains exactly 5 CSV files.
    - A JSON configuration that defines the matching rule for records (see schema below).
    - Optional settings: output folder, degree of parallelism (threads), CSV delimiter, header row present/absent.
2. **File pairing**
    - By default, pair files by **filename** (e.g., FolderA/Orders.csv ⇄ FolderB/Orders.csv). If a file exists in one folder but not the other, report it as "missing file" and continue.
    - (Optional) Add a mode to compare every file in FolderA against every file in FolderB — document which mode is used in logs.
3. **Matching rules**
    - The matching rule must be provided as JSON. It supports:
        - Single-field match (e.g. InvoiceId)
        - Composite match of two fields (e.g. FirstName + LastName)
    - Matching must be case-insensitive by default, with an option to toggle case sensitivity.
    - The JSON also allows specifying trimming/normalization (e.g., trim whitespace).

Example JSONs

json
{ "matchingFields": ["InvoiceId"], "caseSensitive": false, "trim": true }

json
{ "matchingFields": ["FirstName", "LastName"], "caseSensitive": false, "trim": true }

4. **Processing & multithreading**
    - Use multithreading to process file-pairs in parallel.
    - Within a file-pair, large files should be processed using streaming and worker threads (e.g., partition keys or chunked reads) to avoid loading entire files into memory.
    - Respect the configured degree of parallelism.
    - Ensure thread-safety when aggregating results and writing output.
5. **Output**
    - For each file-pair produce:
        - matched.csv — records present in both files (include columns from both sides or unify).
        - only-in-folderA.csv — records present only in FolderA file.
        - only-in-folderB.csv — records present only in FolderB file.
        - reconcile-summary.json — counts: total in A, total in B, matched, only-in-A, only-in-B, processing time.
    - Global summary across all file-pairs (aggregate counts + per-file breakdown).
    - Log file with timestamps, warnings, and errors.
6. **Error handling**
    - Gracefully handle malformed CSV rows (log and skip or write to errors.csv).
    - If CSV headers differ, the tool should not fail: match on the JSON-specified fields; include any missing columns in outputs as empty values.
    - Timeouts/retries are not required but should be logged if a file read fails.
7. **Testing**
    - Unit tests for:
        - Parsing JSON matching rules.
        - Field normalization and comparison (case, trim).
        - Small-file reconciliation correctness.
    - Integration test that runs on a fixture with 5 files in each folder and asserts expected outputs. (optional)
8. **Deliverables**

- Source code (clean, well-documented).
- README with build/run instructions and sample command-lines.
- Sample configuration JSONs.
- Unit and integration tests.
- Example input (small sample CSVs) and expected outputs.

Acceptance criteria

Given two folders each with 5 CSVs and a JSON matching rule, the tool finishes successfully and outputs per-file matched/unmatched CSVs and a global summary.
The tool uses parallel processing to process multiple file-pairs concurrently (verify via logs that multiple tasks run).
Unit and integration tests pass.