**End-to-End Processing System for EvidenceAI**

**Purpose**

The goal of this system is to create a streamlined, user-friendly workflow for processing and managing files across multiple AI tools. This workflow ensures:

* Simplified user interaction with a single directory (input/).
* Automated validation, error handling, and file organization.
* Clear feedback on issues without requiring users to sift through logs.
* Integration with Airtable for lifecycle tracking and metadata management.

**Workflow Overview**

1. **Input Files**: Users place files in the input/ folder. The system automatically detects new files and validates them.
2. **Pre-Processing Validation**: Files are checked for format, integrity, and duplication before processing begins.
3. **Processing**: Valid files are moved to working/, processed, and outputs are saved in done/.
4. **Error Handling**: Errors are logged and summarized in a user-friendly error report with actionable resolutions.
5. **Output Organization**: Outputs are placed in tool-specific folders (claude/, chatgpt/, etc.).
6. **Lifecycle Tracking**: Every file is tracked in Airtable, from input to processing and final outputs.
7. **Automated Cleanup**: Old files and logs are periodically archived or deleted based on retention policies.
8. **Dynamic Configurations**: Users can customize processing rules (e.g., batch size, file size limits) via a configuration file.

**Folder Structure**

bash

Copy code

CopyEvidenceAI/

├── input/ # User-facing folder for new files

├── working/ # Hidden folder for processing

│ ├── temp/ # Temporary processing files

│ ├── logs/ # Logs of processing runs

│ └── errors/ # Detailed error logs

├── processed/ # Archived files with version control

└── done/ # Final results organized by tool

├── claude/ # Claude-ready outputs

├── chatgpt/ # ChatGPT-formatted files

├── notebook/ # NotebookLM optimized files

└── gemini/ # Gemini-specific formats

**Key Features and Reasons**

**1. Pre-Processing Validation**

**What it Does:**

* Ensures only valid, non-duplicate files proceed to processing.
* Checks for:
  + Supported file formats (e.g., PDF).
  + File integrity (e.g., no corruption).
  + Duplicate files (via checksums).

**Why It's Important:**

* Prevents wasted processing time on invalid files.
* Reduces user frustration by catching issues early.
* Ensures consistency and accuracy in outputs.

**2. Batch Metadata Handling**

**What it Does:**

* Generates a batch manifest (batch\_manifest\_<timestamp>.json) summarizing:
  + Files processed, skipped, or errored.
  + Expected outputs.
  + Processing start and end times.
* Syncs this metadata to Airtable.

**Why It's Important:**

* Provides a clear audit trail for each run.
* Allows easy tracking of file lifecycle in Airtable.
* Supports troubleshooting by showing exactly what happened.

**3. Error Handling and Reporting**

**What it Does:**

* Captures errors at each step and categorizes them (e.g., invalid file, processing failure).
* Generates a user-friendly error report (error\_report\_<timestamp>.txt) with:
  + File-specific issues.
  + Error codes and resolutions.
* Displays the error summary in the console after a failed run.

**Why It's Important:**

* Users get actionable feedback without digging through raw logs.
* Minimizes downtime by clearly identifying and resolving issues.

**4. Output Organization**

**What it Does:**

* Saves processed files in done/ organized by tool (e.g., claude/, chatgpt/).
* Ensures outputs are correctly formatted for each tool.
* Includes editable prompt templates for each tool.

**Why It's Important:**

* Simplifies locating outputs for different AI tools.
* Reduces the need for manual adjustments.

**5. Automated Cleanup**

**What it Does:**

* Periodically archives or deletes old files in processed/ and logs/.
* Uses user-defined retention policies in config.json.

**Why It's Important:**

* Prevents the system from being bogged down by old files.
* Frees up disk space automatically.

**6. Dynamic Configurations**

**What it Does:**

* Reads config.json for user-defined rules:
  + Batch size.
  + File size limits.
  + Retention policies.
* Adjusts system behavior dynamically based on these settings.

**Why It's Important:**

* Supports flexibility for different user needs.
* Ensures scalability as processing requirements evolve.

**Implementation Details**

**1. .bat File for Automation**

The .bat file orchestrates the workflow:

batch

Copy code

@echo off

:: Step 1: Load Configuration

python src/load\_config.py config.json

:: Step 2: Validate Files

for %%f in (input\\*.pdf) do (

echo Validating %%f...

python src/validate\_file.py "%%f" airtable\_db\_id

)

:: Step 3: Process Valid Files

if exist working\\*.pdf (

echo Processing valid files...

python src/process\_data.py working\\*.pdf done\

python src/generate\_manifest.py working\ done\logs\

)

:: Step 4: Handle Errors

if errorlevel 1 (

echo Processing failed. Generating error report...

python src/error\_reporter.py working\logs\latest\_run.log

goto END

)

:: Step 5: Cleanup

echo Cleaning up old files...

python src/cleanup.py config.json

:END

echo All tasks completed. Check logs or error report for details.

pause

**2. Airtable Integration**

**Tracked Fields:**

| **Field Name** | **Description** |
| --- | --- |
| File Name | Name of the uploaded file |
| Status | Valid, Invalid, Duplicate |
| Processing Status | Processed, Skipped, Error |
| Batch ID | Identifier for the processing batch |
| Errors | Error details, if any |
| Output Path | Link to processed files in done/ |
| Retention Policy | Days until file is archived or deleted |

**3. Error Explanation and Recovery**

* Common errors are mapped to resolutions in a lookup table:

| **Error Code** | **Description** | **Resolution** |
| --- | --- | --- |
| E001 | Unsupported file format | Ensure files are in PDF format. |
| E002 | Duplicate file detected | File already processed. |
| E003 | Corrupted file | Replace with a valid file. |
| E004 | Processing step failed | Check processing logic. |
| E005 | Output file missing | Ensure write permissions. |

* error\_reporter.py generates a summary report for failed runs, displayed to the user.

**User Flow Summary**

1. **Input:** Place files in input/.
2. **Validation:** The system validates and moves valid files to working/.
3. **Processing:** Files are processed and outputs saved in done/.
4. **Error Handling:** If an error occurs:
   * A report explains what went wrong and how to fix it.
5. **Cleanup:** Old files and logs are automatically managed.
6. **Tracking:** Airtable records the entire lifecycle of every file.

**Next Steps**

1. Implement the described logic in Claude for seamless execution.
2. Create user documentation based on this workflow.