Scriptweaver : Automated Book Publication Workflow

Project Overview

Automated Book Publication Workflow – "Scriptweaver"

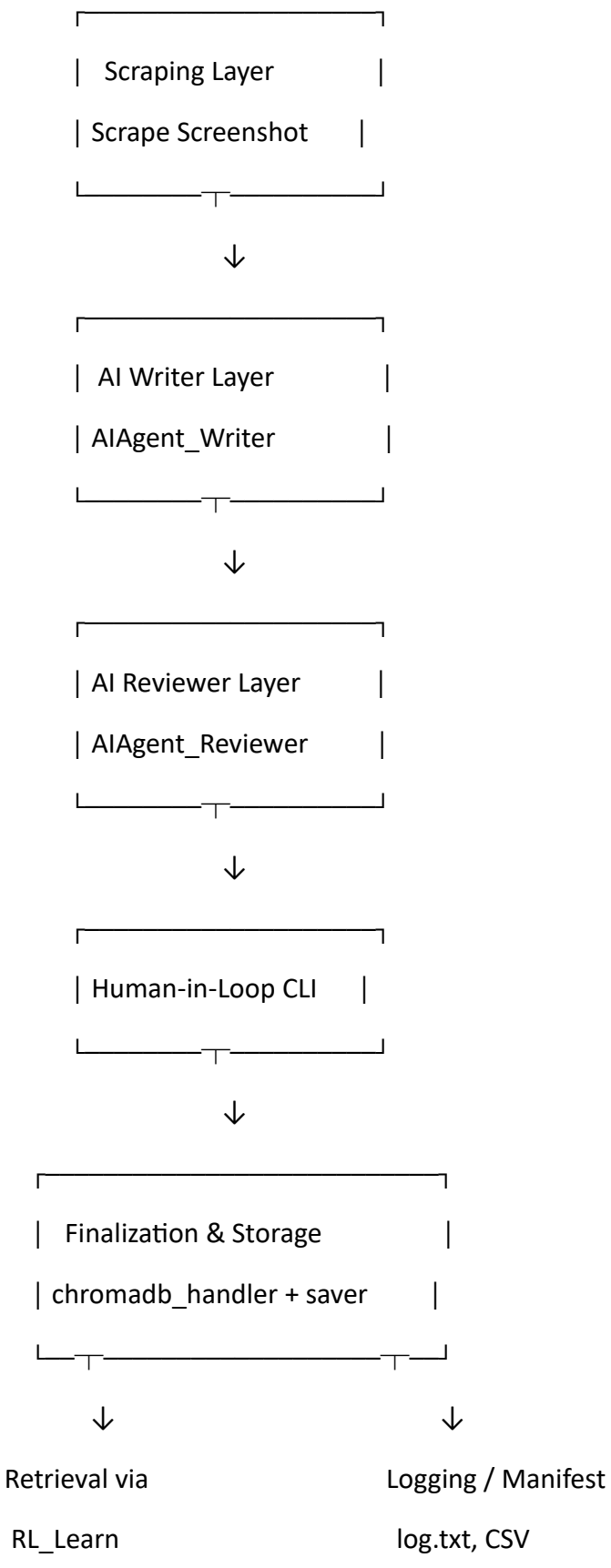
Objective:
Build a pipeline to:

- Scrape raw chapter content
- Apply AI “rewriting” logic
- Refine iteratively with human feedback
- Finalize and store versioned outputs
- Retrieve intelligently using RL-guided search

Core Tools:
Python, Playwright, LLM, ChromaDB, RL Algorithm

High-Level Architecture

System Architecture Diagram



Scraping & Screenshot

Scrape_Screenshot.py

- Uses Playwright to:
    - Load target URL
    - Take full-page screenshot
    - Extract visible text content
- Saves:
    - Text to logs/raw_text/
    - Screenshot to assets/

AI Writing & Reviewing Agents

- **AIAgent_Writer.py**:
    - Rewrites chapter using LLM
- **AIAgent_Reviewer.py**:
    - Refines grammar, tone, and coherence

Human-in-the-Loop Review

Interactive CLI for Human Feedback

- File: Human_Loop.py
- CLI prompt allows:
    - Accept
    - Edit and Save
    - Request re-spin
- Output saved to: logs/final_text/

Finalization & Versioning

ChromaDB Version Storage

- File: chromadb_handler.py
- Stores final version with metadata:
    - chapter01_v3, chapter02_v1, etc.
- Logged into:
    - version_manifest.csv
    - ChromaDB vector store
    - log.txt

Intelligent Retrieval via RL

RL-Learn: Reward-Guided Search

- File: RL_Learn.py
- Uses:
    - retrieve_versions() for vector search
    - Reinforcement logic (epsilon-greedy)
- Maintains feedback in:
    - rl_rewards.json
    - logs/rl_outputs/

Folder Structure & Outputs

Organized Logs & Outputs

- assets/ – Screenshots
- logs/raw_text/ – Scraped chapters
- logs/final_text/ – Final versions
- logs/system/log.txt – Logs
- version_manifest.csv – Metadata record
- chromadb_data/ – Persistent DB

Conclusion

Why Scriptweaver Works

- Modular and agentic
- Supports human-AI co-creation
- Fully versioned with searchable history
- Reproducible and traceable
- Runs locally, no fancy icons or AI signatures
