Draft preview. This draft is a review preview and remains excluded from formal publication outputs. All review workflow examples.

SZD-HTR: A Project-Specific VLM Transcription and Editorial Environment for the Stefan Zweig Estate — A Self-Audit

SZD OCR/HTR Pipeline. https://chpollin.github.io/szd-htr-ocr-pipeline/ (Last Accessed: 2026-08-22). Reviewed by ORCID Christopher Pollin (Digital Humanities Craft OG).

Abstract

This self-audit evaluates SZD-HTR, a project-specific pipeline and editorial environment that creates diplomatic transcriptions of digitised materials from the Stefan Zweig estate with a vision-language model. The assessment covers the public viewer, the local editorial workspace, Page-JSON and archival exports, the trust-tier model, quality signals, correction-derived character error rate, software tests, accessibility features, rights, and sustainability. It examines the public main-branch snapshot at commit `0001f9ea1f1aa40c8839b218798a264f28bac3ed` on 22 August 2026. Eighteen pytest tests passed, and the standalone regression scripts completed 93 distinguishable checks. The catalogue contained 2,452 objects, including 44 with `approved` status and 85 with `agent_verified` status. No object carried `gt_verified`. The audit confirms a substantial, well-documented research workflow with useful provenance fields and interoperable export targets. It also identifies material validity risks. A single page edit can promote an entire object to `approved`, undo and approval removal do not persist consistently, the CER report mixes human and agent changes, one status chart double-counts objects, and export coverage is incomplete. The author is the project lead and domain expert responsible for the reviewed system. This AI-assisted self-audit is a workflow and evidence artifact and has not undergone independent RIDE review.

Scope, Method, and Reviewer Position

SZD-HTR is an experimental sub-project of Stefan Zweig Digital. It combines a Python pipeline for metadata resolution, image-based transcription, verification, and export with a static browser application for catalogue access, facsimile-text comparison, filtering, statistics, and local editorial correction. The system addresses a bounded estate collection held by the Literature Archive Salzburg. Its design is project-specific even where individual formats and modules are reusable.

The author of this review is the project lead and domain expert who defined the requirements, evaluated outputs, and directed development. The repository states that the pipeline architecture, code, frontend, and documentation were generated through AI-assisted Promptotyping. This position provides direct access to design decisions and test artifacts and creates a conflict of interest. The present text therefore reports a self-audit. A formal RIDE publication requires external review and editorial acceptance.

The review text, technical analysis, questionnaire mapping, and architecture diagram were prepared with agentic AI assistance. The named author checked the evidence, answers, claims, and figure and retains scholarly responsibility. This artifact remains an unpublished self-audit and has not undergone independent RIDE review.

The assessed state is the clean public `main` snapshot at commit `0001f9ea1f1aa40c8839b218798a264f28bac3ed`, committed on 21 August 2026 and inspected on 22 August 2026. The review does not incorporate later local modifications. Repository files, committed catalogue data, result objects, schemas, tests, viewer code, and knowledge documents supplied the evidence.

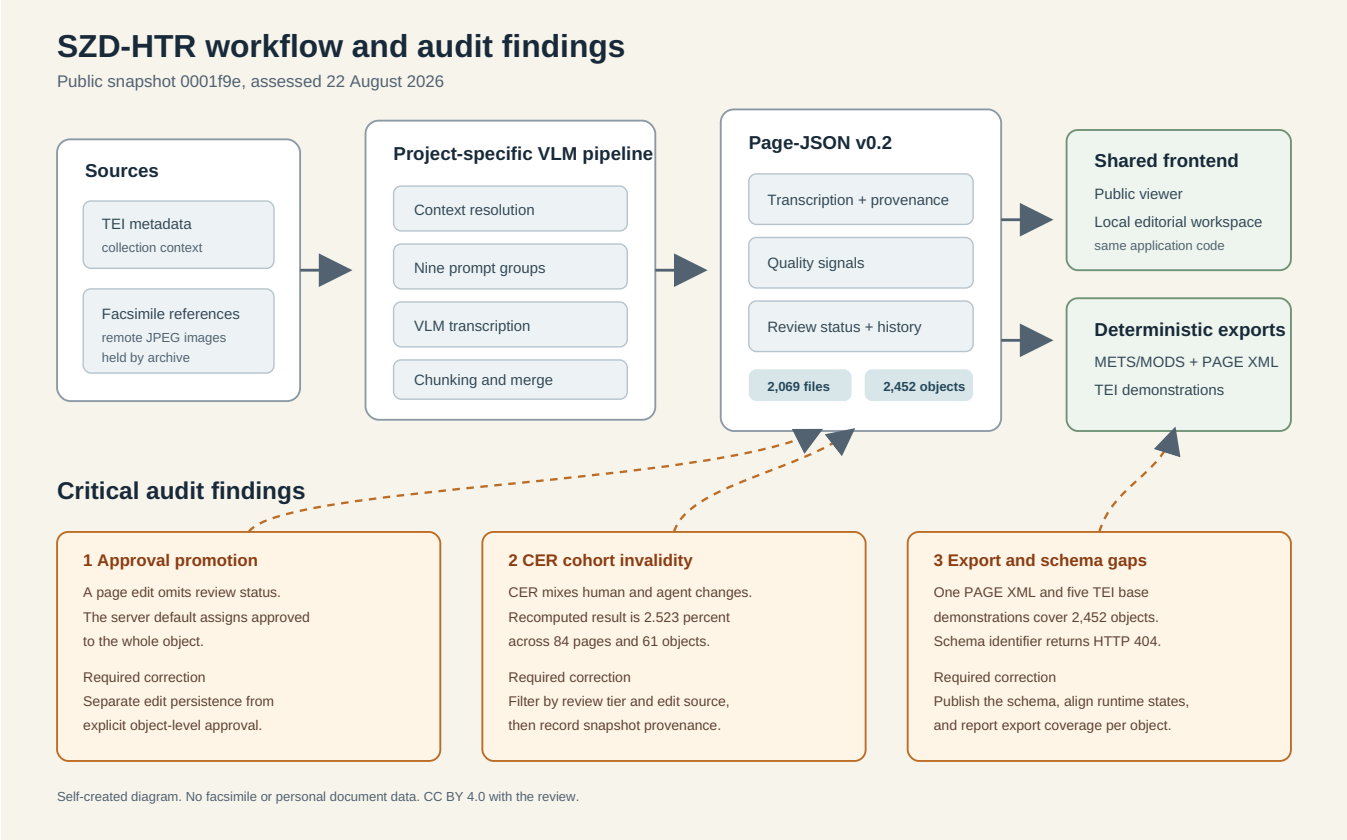
Executable checks comprised the complete pytest invocation documented by the project and the standalone regression scripts under `pipeline/`. All 18 pytest tests passed. The standalone scripts completed 93 distinguishable checks covering quality signals, marker enrichment, deterministic TEI export, canonical collection assignment, and unit derivation. The figures are reported separately because the standalone scripts combine script assertions and unit-test style checks rather than one unified test runner.

The audit did not initiate paid model inference or create new transcriptions. It did not independently establish character-exact ground truth. No facsimile or personal document content is reproduced in this review. The evidence supports claims about the committed system, its stored data, and its executable checks. Claims about transcription accuracy remain limited by the evaluation defects described below.

Purpose and Architecture

The pipeline resolves collection metadata from TEI and backup sources, assigns one of nine document groups, constructs a layered prompt, and sends page images to a configured Gemini vision-language model. The groups cover handwriting, typescript, forms, short texts, tabular material, proofs, convolutes, newspaper clippings, and correspondence. Large objects are divided into chunks before their results are merged.

The principal working representation is Page-JSON v0.2. It separates source metadata, model provenance, page text, optional layout regions, categorical model confidence, deterministic quality signals, evaluation fields, and review provenance. The public static viewer reads collection JSON derived from these results. A local Python server activates editing and review endpoints for the same interface.



The audited SZD-HTR workflow. TEI metadata and repository-hosted facsimile references enter the project-specific VLM pipeline. Page-JSON preserves transcription, provenance, quality signals, and review state. The public viewer and local editorial workspace share a frontend, while deterministic exporters target TEI and METS/MODS with PAGE XML. The highlighted audit findings concern approval promotion, CER cohort validity, and incomplete export coverage. Diagram created for this self-audit and released with the review under CC BY 4.0.

Seven rule-based quality signals inspect properties such as page-image alignment, language consistency, and page-length anomalies. Three signals contribute to `needs_review`, while the others remain informational. These signals support triage. They do not constitute a reliability score and do not alter the stored trust tier.

The public deployment provides a read-only proto-edition. Running the local server exposes an editorial workspace in which corrections are written to result JSON, review status can be assigned, and Git records subsequent changes. The shared frontend reduces duplication between consultation and correction. It also makes the correctness of the local write API central to the validity of the editorial state.

The software is closely coupled to SZD collection identifiers, metadata conventions, prompt groups, repository URLs, and institutional infrastructure. Page-JSON, the export modules, and the trust-tier tests offer reusable components. Reuse of the complete environment in another project requires substantial configuration and code adaptation.

Data Model, Exports, and Interoperability

The pipeline accepts TEI metadata, JSON configuration and result data, and JPEG facsimile references. Its committed outputs include Page-JSON, catalogue JSON, a static HTML viewer, METS/MODS, PAGE XML 2019, and demonstration TEI. UTF-8 is the documented character encoding. The output architecture distinguishes the internal JSON working format from archival XML and downstream TEI.

Coverage is uneven at the assessed snapshot. The catalogue contains 2,452 objects. The repository contains 2,069 Page-JSON files and the same number of METS files. It contains one PAGE XML file. Deterministic TEI export is represented by five base demonstration objects, each accompanied by an enriched variant. These counts show that the exporters exist and produce inspectable artifacts. They do not establish complete archival or TEI export coverage for the catalogue.

The Page-JSON schema supplies an `$id` URL that returned HTTP 404 during the audit. The runtime and review API accept `gt_verified`, while the committed Page-JSON schema does not include that value. This mismatch prevents the schema from describing every state that the application can write.

The TEI exporter is deterministic and covered by regression checks. The PAGE XML and METS architecture follows established exchange layers and is intended for GAMS, Transkribus, eScriptorium, and OCR-D workflows. Evidence for interoperability remains strongest at the structural and test level. Round-trip tests with each named external system are absent from the snapshot.

A complete interoperability proof should publish the Page-JSON schema at its declared identifier, add every writable review state to that schema, and validate all generated Page-JSON during continuous integration. Export coverage should be reported per catalogue object and format. Representative fixtures should pass schema validation and round-trip checks for METS/MODS, PAGE XML, and TEI.

The local server exposes documented endpoints for editing, approval, status, and Git state. These endpoints support the bundled frontend and are covered partly by trust-tier tests. The repository declares no stable public API version or compatibility policy. External integration should therefore use the file formats rather than rely on the current local endpoint contract.

Trust Tiers and Editorial State

Four states are stored in the project data. `gt_verified` denotes character-exact human verification, `approved` denotes human expert review, `agent_verified` denotes image-text comparison by a vision agent, and absence of a review block denotes unreviewed machine transcription. The display groups the first two states as human-checked. At the snapshot, 2,323 catalogue objects were unreviewed, 85 were agent-verified, 44 were approved, and none was ground-truth verified. Among the unreviewed objects, 324 carried the `needsReview` triage flag.

The data model keeps review tier, categorical model confidence, and deterministic quality signals separate. This is a sound epistemic design. It prevents a heuristic signal or model self-assessment from being presented as equivalent to human source checking.

The local edit path violates that separation. The frontend function `saveCurrentEdit()` submits `/api/edit` without a review status. The server handler defaults a missing status to `approved`. Saving one corrected page can therefore label the entire object as human-approved even when its other pages have not been checked. The initial machine transcription and edit history are preserved, but the object-level status overstates the completed review.

Reversibility is also asymmetric. Undo, discard, and approval removal update browser state without consistently reverting the persistent result JSON. Network and API failures are caught without a visible error in several write paths. The interface can consequently report a local state that differs from the file on disk.

The edit and approval transitions should be separated. Page edits require an explicit draft or edited state and must never infer object-level approval. Approval should use a dedicated endpoint that verifies the intended scope and reports the stored result. Undo, discard, and status removal require persistent API operations. Failed writes must remain visible until the user retries or resolves them. Tests should cover every transition against multi-page objects and reload the saved JSON before asserting the resulting status.

CER, Metrics, and Dashboard Validity

The project preserves the first machine transcription in `transcription_llm` and stores subsequent changes in page-level `edit_history`. The CER report compares the preserved machine text with the current text. Recomputing the report from the assessed snapshot produced 84 pages from 61 objects, 132,154 reference characters, and an aggregate CER of 2.523 percent. The committed report described 56 pages and 0.962 percent.

The reporting script filters neither the object review status nor `edit_history.source`. Human and agent changes can enter the same calculation. The result cannot be interpreted as character error rate against independently established human ground truth. The difference between the committed and recomputed reports also shows that the report is stale relative to the committed data or that its selection basis changed without recorded provenance.

The dashboard contains a second aggregation defect. The review-status donut applies overlapping predicates and counts 35 objects in more than one slice. Other catalogue summaries use mutually exclusive precedence and produce the 2,452-object partition reported above. Status visualisations should derive from one canonical classifier and assert that every object contributes to exactly one tier.

A defensible CER artifact needs an explicit cohort selected by human review status and human-sourced corrections. It should record the data commit, script commit, normalization profile, object and page identifiers, character total, aggregation method, and confidence interval. Agent corrections can be reported as a separate diagnostic cohort. The committed report and dashboard data should be regenerated in a checked build so stale metrics cannot reach the viewer.

Interface, Usability, and Accessibility

The viewer offers catalogue search, collection and quality filters, object navigation, facsimile-text comparison, statistics, model-consensus displays, and a project-transparency section. The local workspace adds page editing, approval actions, reviewer attribution, Git status, and curation progress. The interface exposes the pipeline state in a form that humanities researchers and digital editors can inspect without reading result JSON directly.

Accessibility features are present in the implementation. The HTML includes a skip link, language metadata, labelled controls, live regions for changing status, alternative text for the facsimile image, and keyboard shortcuts for central viewer operations. These features justify an affirmative questionnaire answer about implemented accessibility support. The repository contains no formal Web Content Accessibility Guidelines audit, screen-reader test, keyboard-only test, zoom and reflow test, or documented contrast assessment.

Supported browser and operating-system matrices are not documented through repeatable compatibility tests. The questionnaire records the desktop and laptop working context and the browser used for the documented web workflow. Deployment platforms remain unanswered because the snapshot does not provide platform-specific support evidence. Mobile and tablet suitability should be established through task-based testing rather than viewport screenshots alone.

Write operations require clearer feedback. A saved, approved, reverted, or failed state should be confirmed from the server response and reflected consistently after reload. The current silent catches reduce user confidence in editorial persistence and make status errors difficult to detect during routine correction.

Rights, Data Governance, and Sustainability

The repository licenses pipeline code, viewer code, and tooling under MIT. It licenses documentation and other textual project content under CC BY 4.0. Transcriptions and digitised facsimiles derive from the Stefan Zweig estate and remain subject to the rights of the holding archive. The self-created diagram in this review contains no source image or personal document data and follows the review licence.

Rights and provider governance remain incomplete at the institutional level. The documentation identifies the holding archive, the external model provider, and the technical data flow. It does not establish the institutional controller and processor roles, approved retention terms, contractual basis, or publication licence for derived transcription data. These decisions are required before a production workflow can make general reuse claims.

The setup uses an environment variable for the model API key, and the public viewer does not require that key. The software is free and open, while productive model inference can incur external provider charges. The security documentation records system-specific risks. Dependency reproducibility remains partial because the snapshot has no lockfile and combines exact versions with minimum-version constraints.

The public repository has no tagged release, archived software DOI, continuous-integration workflow, or machine-checked lockfile. It does contain extensive Markdown documentation, a public issue tracker, modular pipeline scripts, schemas, and executable tests. Long-term support is stated through ongoing project activity rather than a release and maintenance policy.

The README discloses AI-assisted implementation and names the project lead's role. Git records individual contributions. The project lacks a consolidated contributor and role statement, citation file, and formal citation guidance. The questionnaire therefore records contributor acknowledgement and citation support as negative findings.

A sustainable release should bind a tagged source snapshot, environment lock, schema set, generated metrics, test results, and documentation to one version. Archiving that release under a persistent identifier would make later comparison possible. Continuous integration should execute schema validation, the unified test suite, export-coverage checks, and dashboard invariants.

Assessment and Priorities for Improvement

SZD-HTR demonstrates a coherent workflow for a large and heterogeneous archival collection. Its layered prompts, explicit Page-JSON provenance, separate quality signals, facsimile-linked viewer, local editorial workspace, and archival export targets are substantial research-software achievements. The public repository makes the architecture and many decisions inspectable.

The highest-priority correction concerns editorial truth claims. Saving text and approving an object must become separate, explicit, persistent actions. The system should prevent any partial page edit from promoting an entire multi-page object. Server-confirmed state and visible errors are necessary for reliable expert review.

The second priority concerns measurement validity. CER needs a reproducible human-ground-truth cohort, provenance-aware filtering, and regeneration from the exact published snapshot. Dashboard status counts need mutually exclusive classification and invariant tests.

The third priority concerns interoperability evidence. The Page-JSON schema must resolve at its declared identifier and include every runtime state. Export coverage should be complete or explicitly delimited, and representative outputs should pass round-trip tests with the claimed target systems.

Accessibility and data governance require documented verification. A formal accessibility audit should cover keyboard operation, assistive technology, contrast, zoom, reflow, and responsive task completion. Institutional agreements should define the rights and provider conditions for source images, prompts, model outputs, corrected transcriptions, and public derivatives.

At the assessed snapshot, the system is suitable for experimental internal transcription workflows. Its viewer technically supports public exploration of machine-generated text where the holding archive has documented the publication rights, personal-data clearance, and provider conditions for the selected material. Use as an authoritative record of human verification requires the trust-tier fix. Quantitative accuracy claims require a corrected and reproducible evaluation pipeline.

Independent evaluation should repeat the editorial workflow on a stratified sample of document groups, inspect saved state after reload, validate representative exports, and compare the reported CER cohort with the underlying correction provenance. Those checks would provide the evidence required for a formal RIDE review.

References

Institute for Documentology and Scholarly Editing. 2018. “Criteria for Reviewing Tools and Environments for Digital Scholarly Editing, version 1.0.” https://www.i-d-e.de/publikationen/weitereschriften/criteria-tools-version-1/.

SZD OCR/HTR Pipeline. 2026. Public source snapshot `0001f9ea1f1aa40c8839b218798a264f28bac3ed` . https://github.com/chpollin/szd-htr-ocr-pipeline/tree/0001f9ea1f1aa40c8839b218798a264f28bac3ed.

PAGE XML. 2019. “PAGE XML Schema 2019-07-15.” https://github.com/PRImA-Research-Lab/PAGE-XML.

Figures

The audited SZD-HTR workflow. TEI metadata and repository-hosted facsimile references enter the project-specific VLM pipeline. Page-JSON preserves transcription, provenance, quality signals, and review state. The public viewer and local editorial workspace share a frontend, while deterministic exporters target TEI and METS/MODS with PAGE XML. The highlighted audit findings concern approval promotion, CER cohort validity, and incomplete export coverage. Diagram created for this self-audit and released with the review under CC BY 4.0.

Drafted: 2026-08-22 · build cb141fb (2026-08-23T11:45:27+02:00)