# teiCrafter: A Browser-Based TEI XML Editor with Optional LLM Support — A Self-Audit

*teiCrafter*. https://digitalhumanitiescraft.github.io/teiCrafter/ (Last Accessed: 2026-08-22). Reviewed by ORCID Christopher Pollin (Digital Humanities Craft OG).



## Abstract

This draft self-audit evaluates teiCrafter, a client-only TEI XML editor that runs in a web browser and provides optional large-language-model support for initial TEI generation and annotation proposals. The assessment applies the RIDE criteria for tools and environments to a locally tested development state and to two source objects from the Zurich Central Library's OCR-to-TEI workflow for a digital edition of writings by and about Jeanne Hersch. The principal developer authored the review; AI-assisted agentic systems supported drafting, questionnaire mapping, and preparation of the workflow diagram. Browser and automated checks provide evidence for data preservation, local processing, project-specific configuration, and reviewable AI proposals. The tests also establish that the working tree is not a reproducible release, integrated project-schema validation and internal version history are absent, browser portability is incompletely documented, accessibility has not been audited formally, and AI quality depends on the selected model and task. This artifact has received neither independent review nor editorial approval for publication.

## Scope, Method, and Reviewer Position

teiCrafter describes itself as a browser-based, client-only, lossless editor for arbitrary TEI XML. Its deterministic editing core is the primary subject of this audit. The scope also covers two optional LLM-assisted entry points, which generate an initial TEI document from plain text or propose annotations for the current folio, and the ingestion of the resulting review bundle into the static RIDE preview workflow. External usability, production readiness, mobile use, and formal accessibility conformance remain outside the evidence established here.

The author is also the principal developer of teiCrafter. This developer self-audit has received neither independent review nor editorial approval and makes no claim to publication as a RIDE review. AI-assisted agentic systems supported preparation of the review text, questionnaire mapping, technical analysis, and the

workflow diagram. The author revised and checked their output against repository inspection, executable tests, and recorded browser interactions and remains responsible for every assessment. Judgements about usability and scholarly value remain provisional until independent users repeat the workflow.

The tested state was the local working tree on 22 August 2026. Its baseline was Git commit `f57db14161195ed1d5131ab83afb945aba787a42`, supplemented by uncommitted hardening changes produced during the audit. The exact tested state therefore cannot be reconstructed from that commit alone, and the public GitHub Pages deployment may differ. Recorded browser interaction covered two real source files from the Jeanne Hersch project, numbered 1000 and 1540. Automated checks on the same working tree comprised 52 passing JavaScript proofs, syntax checks for all changed JavaScript modules, a harness self-test with 14 passing cases, and four synthetic validation fixtures with a score of 100. The Hersch TEI files and facsimiles are treated as rights-restricted research data and are excluded from this bundle.

## Tool Identity and Editorial Tasks

teiCrafter is research software in the form of a static web application. Users can open the hosted editor without installing an application or creating an account. The browser reads local TEI files and, where supported, writes changes back through the File System Access API. A download path remains available when direct file-system access is unavailable. The core editor operates without a server, user account, telemetry service, or language model.

The principal tasks are transcription correction, line- or word-level editing, annotation, authority linking, index maintenance, facsimile consultation, and inspection of the underlying XML source. The document determines the editing unit. Tokenized TEI is handled at word level, while documents structured primarily through line breaks are handled at line level. This makes the default path usable across heterogeneous TEI documents without an initial project configuration.

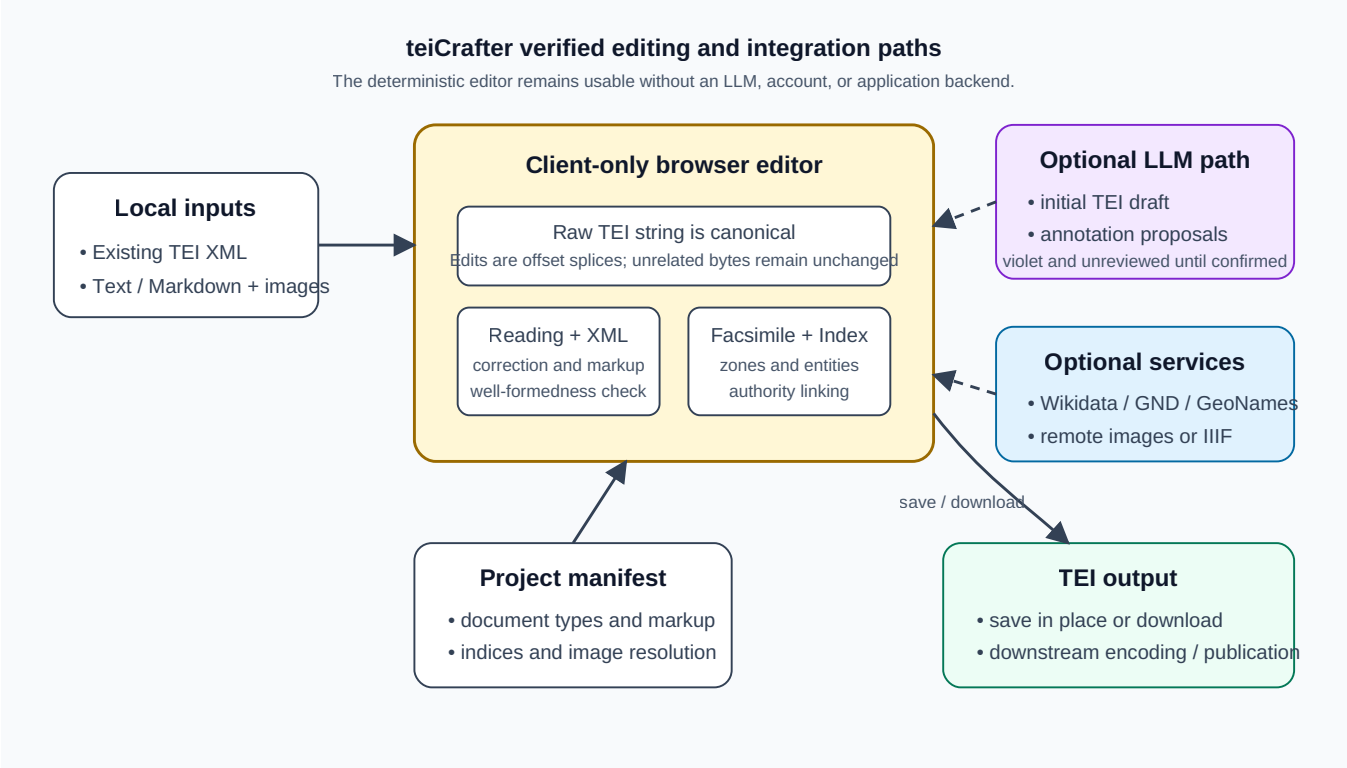
An optional `teicrafter.project.json` manifest adds project-specific document types, allowed markup, indices, authority services, facsimile resolution, and LLM instructions. The manifest is significant for edition workflows because editorial rules remain explicit project data. It also limits the claim of complete generality; any TEI document can be opened, but productive project-specific annotation depends on a suitable manifest or built-in profile.

The application is positioned between transcription or HTR output and more extensive encoding or publication systems. That position is descriptive rather than exclusive. teiCrafter itself is a TEI XML editor. Users can create a deterministic TEI draft from text, modify existing XML, add structural and semantic markup, manage stand-off entities, and inspect source. Its narrower emphasis is efficient correction and enrichment with immediate facsimile context while preserving the original file representation.

## Architecture, Data Model, and Interoperability

The application uses HTML, CSS, and native JavaScript modules without a build step or application framework. OpenSeadragon provides the deep-zoom facsimile viewer, and a vendored TEI P5 compilation supports element guidance. The raw XML string is the canonical document. Edits are applied as offset-based string splices, and saving returns that string. An untouched document is therefore reproduced byte for byte, while a changed document retains unrelated whitespace, attribute order, and entity notation. This design makes ordinary file diffs suitable for reviewing editorial interventions.

Input comprises TEI XML, plain-text and Markdown files, and page images for the deterministic text-and-images on-ramp. Output is TEI XML. Text is handled as UTF-8 in the documented workflow. The plain-text converter deliberately performs transport without semantic interpretation. Blank lines create paragraphs, line breaks remain explicit, and page markers can connect text and images. The application does not currently perform Relax NG or Schematron validation against a project schema. Its live checks cover XML well-formedness and the lossless editing invariant, while project manifests constrain the interface vocabulary.




The verified teiCrafter workflow. Local TEI or deterministic text conversion enters the lossless browser editor; project manifests add editorial rules and facsimile resolution; optional LLM output enters only as unreviewed material; the saved TEI remains compatible with downstream edition systems.

Interoperability rests primarily on TEI P5 input and output and on the preservation of existing XML rather than transformation into a proprietary internal format. The editor can resolve local page images, project-specific image paths, and IIIF-compatible sources. Authority reconciliation is available for Wikidata, GND, and GeoNames. These requests are direct browser requests and require network access; the editor remains functional without them.

Trackability is partly external. Machine-generated proposals carry `resp="#ai"` and remain visually marked until a person confirms or rejects them. Rejected proposals and session-created AI responsibility metadata are removed when the original state is restored. Human edits, however, are not stored as an internal version history or audit log. Reproducibility therefore depends on saving files deliberately and using an external version-control or repository workflow.

## Worked Use Case with Jeanne Hersch Project Material

The audit tested teiCrafter against two source objects from the Zurich Central Library's OCR-to-TEI workflow for a digital edition of writings by and about Jeanne Hersch. The objects exercise the line-based profile with different page counts and stand-off structures. Document 1000, a 1973 periodical issue titled *Transformer l'école ou la supprimer?*, opened with four pages; document 1540, *Mein Judentum* from 1978, opened with eight. In both cases the editor recognized the exact project signature, connected lines to facsimile zones, exposed persons and other index data, and retained the project's inline GND representation through save and reopen.

The first use case covered the complete deterministic sequence from opening the TEI and attaching the local facsimile folder in read-only mode through page navigation, zone inspection, text correction, index inspection, saving, downloading, and reopening the result. The second use case added page switching, annotation, and a live GND candidate lookup for Jeanne Hersch. The adapter translates inline authority information into the editor's register model while the document is open and restores the project's established inline representation on save. Mention-level provenance attributes such as source, certainty, and responsibility survived the round trip.

This result matters because the generic editor model and the Hersch project model are not identical. A direct normalization into teiCrafter's preferred stand-off representation would produce avoidable source changes. The boundary adapter instead treats the existing TEI convention as authoritative. Its current detection is intentionally exact. The built-in profile activates only for `TEI/@type="naegeli"`. Other Hersch documents or future schema variants require a manifest or an explicit extension.

The facsimile workflow also exposed a data-governance boundary. A separately attached image folder is read-only and its files are neither copied into the project nor persisted by ordinary Save. Images added through the deterministic text-and-images on-ramp follow a separate path; a project-folder save writes those files next to the TEI. The distinction prevents a consultation source from being redistributed through an ordinary save operation and requires the interface to identify the active image path clearly. No Hersch source text or facsimile is reproduced in this bundle. Figure 1 is an original schematic prepared for this audit and is covered by the draft's Creative Commons Attribution 4.0 licence.

## Optional LLM Assistance

LLM functionality is optional at both the build and user levels. A runtime toggle hides the AI entry points while leaving the deterministic editor intact. Users choose a provider and model; local Ollama model identifiers are accepted exactly, while cloud providers use maintained model catalogues. API keys remain in module-scoped memory for the browser session. Prompts and folio text are transmitted directly to the selected provider only when the user submits an AI action.

The generation path accepts a plain-text source and requests a complete TEI P5 document. Before the response enters the editor, a structural gate requires well-formed XML, the TEI namespace, a minimum `fileDesc`, and `text/body`, and rejects a document type declaration. Repeated trials with a local Mistral model failed this gate because the model omitted required header content. A controlled valid response passed and opened as a marked, unreviewed TEI document. The gate therefore prevents malformed documents from entering the editor but cannot establish the semantic correctness of accepted markup.

The proposal path sends the current folio text together with the active annotation vocabulary and project instructions. Returned spans must match the source exactly or resolve through a unique case-insensitive match. Accepted entity, markup, textual-criticism, and note proposals are inserted with AI responsibility metadata and shown in violet. Inline proposals and stand-off notes each expose confirm and reject controls. This state model is clearer than placing model output directly into an apparently finished document.

The local Mistral trial also established the main limitation. Proposals were structurally reviewable but semantically unreliable and included hallucinated interpretations. Model choice, prompt design, language, historical spelling, and local context all affect quality. teiCrafter currently provides no benchmark score, confidence calibration, or project-specific quality threshold. Its defensible contribution is the review boundary around model output. Editorial acceptance remains a scholarly decision made by the user.

## Usability, Documentation, and Sustainability

The interface is a two-pane workbench. The left pane switches between a diplomatic reading view and editable XML source; the right pane switches among facsimile, index, and project views. Page controls, document facts, integrity status, annotation actions, and AI review states remain visible near the object they affect. In the developer-led Hersch test, the sequence from page image to line correction and index entry was coherent, and the interface density remained manageable. Independent usability testing is required to establish whether first-time users can understand the distinction between reading units, XML source, project manifests, and save modes without developer guidance.

The hosted application requires no installation. Local development requires a static HTTP server because ES modules and file-system features should not be run from a `file://` URL. Direct folder opening and save-in-place depend on the File System Access API and are documented as Chromium-specific. A previous partial interaction check used Google Chrome, and the current Hersch run used the in-app Chromium browser; these environments account for the questionnaire selections "Google Chrome" and "Other." Feature parity across Firefox and Safari was not established. Mobile use was not evaluated, and the desktop two-pane interface is the intended working environment.

Documentation is extensive for a research preview. The repository includes a README with quick-start instructions and worked cases, a security and data-handling statement, software citation metadata, an MIT licence, contribution guidance, and a structured knowledge base describing architecture, data, design, integration, specification, and testing. GitHub issues provide a public support and bug-reporting channel; security reports have a private email route. The repository is modular and inspectable, although it does not expose a stable public programming API or a backward-compatibility policy. The questionnaire therefore records ease of code analysis as supported and ease of extension as unknown.

The automated proof suite covers parsing, offset preservation, annotation operations, project profiles, facsimile-folder behavior, Hersch interchange, inline GND reopening, LLM prompts, proposal application, generation gates, and worked examples. These checks provide direct evidence for the lossless core. External usability studies, cross-browser testing, security review, and a formal accessibility audit remain open. The markup contains ARIA roles and labels, visible focus treatments, keyboard interaction for central controls, and reduced-motion handling; conformance with the Web Content Accessibility Guidelines has not been established.

Sustainability signals are mixed. The code is open under MIT, hosted in a public Git repository, documented for reuse and adaptation, and supplied with `CITATION.cff` and CodeMeta. The maintainer states that the tool is used in internal and funded projects. At the same time, the software is explicitly a research preview, has no tagged stable release, version number, release date, or software DOI, and makes no production-readiness claim. Long-term maintenance currently depends on a small development context.

## Assessment and Priorities for Improvement

teiCrafter already realizes its core aim of local TEI editing with minimal unintended document change. The offset-based model, visible XML source, facsimile-linked reading surface, project manifests, and reviewable AI state address recurring problems in digital edition production. The Hersch case demonstrates that the editor can integrate with a project-specific TEI convention while preserving the established authority representation.

The most important improvements concern verification and operational clarity. Project schemas should be connectable to explicit Relax NG and Schematron validation so that well-formed output can also be tested against editorial constraints. The save interface should continue to distinguish download, project-folder save, read-only facsimile attachment, and newly imported images. Browser support should be documented feature by feature. A formal accessibility audit and keyboard-only use test are required before making accessibility claims beyond implemented interface features.

For LLM-assisted work, evaluation should move from individual demonstrations to task-specific fixtures and expert-labelled benchmarks. Model, prompt, provider, timestamp, and accepted or rejected proposals should be exportable as provenance when a project requires reproducibility. The current visual and structural distinction between proposed and confirmed content is a sound basis for that work. The deterministic editor should remain independently usable because it is the more mature and broadly defensible part of the tool.

For software sustainability, the next public milestone should define a versioned release, archive it under a persistent identifier, state the supported browser matrix, and connect documentation and tests to that release. Independent review by editors who are not involved in development is necessary before a formal RIDE submission. The available evidence supports describing teiCrafter as a substantial research prototype with a validated lossless editing core and demonstrated integration with the Hersch project format. The semantic performance of its LLM features remains unquantified and experimental.

---

## References


Sichani, Anna-Maria, and Elena Spadini, in collaboration with the members of the IDE. 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/.

Pollin, Christopher, Franz Fischer, Patrick Sahle, Martina Scholger, and Georg Vogeler. 2025. “When it was 2024 — Generative AI in the Field of Digital Scholarly Editions.” *Zeitschrift für digitale Geisteswissenschaften* 10. https://doi.org/10.17175/2025_008.

Pollin, Christopher, Christian Steiner, and Constantin Zach. 2023. “New Ways of Creating Research Data: Conversion of Unstructured Text to TEI XML Using GPT on the Correspondence of Hugo Schuchardt with a Web Prototype for Prompt Engineering.” FORGE 2023. https://doi.org/10.5281/zenodo.8425163.

TEI Consortium. 2026. “TEI P5: Guidelines for Electronic Text Encoding and Interchange.” https://tei-c.org/guidelines/p5/.

## Figures

The verified teiCrafter workflow. Local TEI or deterministic text conversion enters the lossless browser editor; project manifests add editorial rules and facsimile resolution; optional LLM output enters only as unreviewed material; the saved TEI remains compatible with downstream edition systems.

---



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