*Computational Humanities Research*

**www.cambridge.org/chr**

# RUCAI: An open-source, local-first AI teaching assistant for course planning and classroom support

Frederik Henriksen

Department of Communication and Arts, Roskilde University, Denmark

**Software Paper**

**Cite this article:** Henriksen Frederik. 2026. "RUCAI: An open-source, local-first AI teaching assistant for course planning and classroom support." *Computational Humanities Research*, **2**:e16, https://doi.org/10.1017/chr.2026.10038

Received: 7 March 2026 UTC
Revised: 20 May 2026 UTC
Accepted: 9 June 2026 UTC

**Keywords:**
course planning; digital humanities; educational technology; local-first software; retrieval-augmented generation

**Corresponding author:**
Frederik Henriksen;
Email: frmohe@ruc.dk

## Abstract

RUCAI is an AI-based open-source software platform designed for humanities and social sciences educational workflows that require traceability to assigned course materials. RUCAI has a teacher-only workspace in which teachers can curate class exercises grounded in curriculum documents (articles, book chapters, and notes). From here, the teacher can set up workspaces grounded explicitly in curriculum materials, learning outcomes, and course requirements for students to use throughout the course. The system is set up to run on institution-controlled infrastructure using FastAPI, PostgreSQL, and a local LLM model serving via Ollama, and is intended to be deployable on a GPU-enabled virtual machine. RUCAI's primary contributions are practical: (1) it presents a software architecture for local-first, source-grounded educational AI that treats the course as the primary software object; (2) it documents concrete implementation choices for reliable ingestion and retrieval across heterogeneous course materials, including a three-tier extraction pipeline, word-based chunking, and intent-aware retrieval; and (3) it describes a publish-to-student mechanism that lets teachers freeze course context and prompt constraints into student-facing instances. The paper also reports pilot observations, current limitations, and future development directions.

## Plain language summary

Teachers increasingly encounter AI tools that can write fluent text but cannot clearly show where their answers come from, and institutions often cannot send course materials to external cloud services. RUCAI is a locally deployable teaching copilot that helps teachers work with their own course descriptions and readings. Teachers upload course documents, ask questions for planning or teaching support, and receive answers that point back to the specific course texts used as evidence. RUCAI also lets teachers publish restricted student-only chat workspaces so students can ask questions only within the course materials and instructions the teacher has chosen. RUCAI is designed for teaching contexts where the primary scholarly activity is the interpretation of assigned texts – a pattern characteristic of humanities disciplines and text-centered social sciences.

## Introduction

There is growing recognition that AI, including generative AI, may offer benefits for learning and teaching within higher education (UNESCO 2024). Large language models are increasingly used in university teaching contexts (Crompton and Burke 2023; Leitgeb and Leitgeb 2025), but much current tooling for teaching support remains difficult to align with core humanities requirements: transparent grounding in course materials, explicit uncertainty when evidence is weak, and institutional control over data and model operations. In practice, many teachers need support that is neither a generic chatbot nor a full learning management system (LMS). They need a workflow-centered assistant that can ingest their own course descriptions and assigned texts, answer planning questions with explicit source references, help formulate exercises, and make these capabilities available to students in constrained forms.

RUCAI[1] is a local-first teaching system designed for this gap. It is built as a teacher-grounded retrieval-augmented generation (RAG) copilot that runs on institution-controlled infrastructure and prioritizes source discipline. RUCAI supports immediate ingestion of course documents (currently PDF and DOCX), asynchronous indexing, grounded question answering, persistent

[1]RUCAI (pronounced/ru:kaI/) originated as a working name combining the author's institutional abbreviation with "AI." The name is intentionally kept as a project identity rather than an institutional label – like the Hokkaido pumpkin it phonetically resembles, the system is meant to be robust, adaptable, and useful across different growing conditions.

This article was awarded Open Data and Open Materials badges for transparent practices. See the Data availability statement for details.

© The Author(s), 2026. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (https://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

course-scoped chat history, and publication of student-facing chat workspaces – "student instances" – derived from a teacher-configured course state. Student instances are implemented as frozen runtime snapshots: they capture the effective system prompt, course description, and chunk corpus at publication time, and provide access-scoped chat without exposing teacher-side configuration. Constraints on student interaction are enforced through corpus scoping (retrieval is limited to the instance's document set) and prompt governance (uncertainty rules are locked in the system prompt layer).

Rather than being centered on generic prompt interaction or custom GPTs, RUCAI is centered on course workflows: defining course context, ingesting literature, querying across texts, and optionally exposing a constrained student interface (Brown, Roman, and Devereux 2025; Lewis et al. 2020). RUCAI is developed as open-source software, with source code available through a public GitHub repository and archived on Zenodo. The target audiences are: (1) teachers and course planners in humanities and social sciences, (2) research software engineers and digital humanities (DH) practitioners building local AI workflows, and (3) institutions that require privacy-preserving deployment and transparent source attribution.

RUCAI is currently developed as a pilot-oriented system with a strong emphasis on reproducibility in local VM settings and clear operational controls. This paper makes three contributions:

1. It presents a software architecture for local-first, source-grounded educational AI assistance that combines teacher workflow design with operationally simple deployment.
2. It documents concrete implementation choices for reliable ingestion and retrieval in heterogeneous course materials, including a three-tier extraction pipeline (pypdf → docling → MinerU), word-based chunking, and custom language detection.
3. It outlines a publish-to-student mechanism that lets teachers freeze course context and prompt constraints into student-facing instances.

## Context

RUCAI was developed to address a recurrent problem in course preparation and delivery related to how teachers need to think across multiple assigned texts while maintaining interpretive accountability. Existing AI interfaces often produce fluent but weakly grounded responses and provide little control over what textual evidence is being used. In humanities teaching, this is not a minor issue. The pedagogical value of AI assistance depends on whether claims can be traced back to course literature and whether uncertainty is represented clearly (Qi et al. 2024; Wallat et al. 2024). Recent empirical evidence from humanities education confirms that while generative AI can support initial familiarization with course material, it lacks the depth required for advanced study unless combined with authoritative, source-grounded resources (Crompton and Burke 2023; Wessels 2025).

Two context drivers shaped RUCAI's design. *First*, course planning is iterative and distributed across artifacts. Teachers move between course descriptions, institutional learning objectives, assigned readings, and exercise design. RUCAI, therefore, treats a course as a first-class object, with an explicit course title, course description, and course-specific prompt instructions. This design supports continuity across sessions and avoids the stateless restart pattern common in ad hoc chatbot use. *Second*, institutional and legal constraints shape what software universities can deploy. European universities operate under the General Data Protection Regulation (GDPR), which restricts processing of student and course data on servers outside institutional control. Data sovereignty concerns are heightened by the dominance of US-based cloud providers – whose three largest firms control about 70% of the EU cloud infrastructure market – and by access provisions under US legislation such as the CLOUD Act (European Parliament 2025). These constraints mean that routing course materials through proprietary AI services (e.g., ChatGPT, Copilot) may not be permissible without institutional risk assessment. In many educational settings, the default assumption cannot be unrestricted cloud processing of course materials. RUCAI is designed to run locally on controlled VM infrastructure with local inference via Ollama, relational storage in PostgreSQL, and vector search via pgvector. This local-first orientation supports data governance requirements while retaining operational flexibility (Huang and Lee 2025; Karakurt and Akbulut 2026; Koeser et al. 2025).

The methodological orientation supports the expectation that AI-assisted responses should be traceable to specific course texts and should express uncertainty when evidence is limited. RUCAI enforces source-grounded prompts, requests explicit citation markers, and returns source metadata to users. When context is insufficient, the intended behavior is not extrapolative confidence but explicit uncertainty (Pollin et al. 2025).

These terms – close reading, hermeneutic interpretation, interpretive accountability – are invoked in this paper with a specific and limited software-level meaning. RUCAI does not simulate hermeneutic interpretation, nor does it perform close reading in the humanistic sense, leaving this to student–teacher interactions. Rather, it operationalizes three features that make humanistic interpretive practices tractable when working with AI assistance:

1. Source traceability: Responses include explicit references to the specific course texts used as retrieval evidence, grouped by source identity (not by chunk). This supports the humanistic expectation that claims are attributable to primary texts rather than to an opaque model.
2. Uncertainty marking: When retrieved evidence is insufficient, the system is instructed – through a locked prompt rule – to express this explicitly rather than extrapolate. This is a software-level proxy for the close-reading norm of not overstating what a text says.
3. Evidence scope control: Retrieval is limited to the teacher's indexed course materials. The system does not retrieve external web content, and its prompts instruct the model to base answers on the uploaded corpus. This corresponds to the practice of constraining interpretation to assigned texts within a course context. The scope is intentionally conservative, but not static: throughout a course, the teacher can update the system prompt to reflect course developments, class conversations, comments, objections, and emerging interpretations. RUCAI therefore treats the uploaded corpus as the evidence boundary for generated answers, while allowing the teacher to adjust the instructional framing around that corpus as the course develops. This is not presented as the full boundary of human interpretation.

These three features do not constitute a humanistic method; they are minimal software conditions for AI assistance that do not

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

**Table 1.** Feature comparison: RUCAI and selected local RAG systems

| Feature | RUCAI | AnythingLLM | Proprietary chatbots (e.g., ChatGPT, Copilot) | Generic learning management systems (LMS) |
|---|---|---|---|---|
| Local-first deployment (no cloud API required) | ✓ | ✓ | ✗ | Vendor-dependent |
| Workspace-level document scoping | ✓ | ✓ | Limited (file upload, session-scoped) | Varies |
| Course as first-class software object (title, description, activation state) | ✓ | ✗ | ✗ | ✗ |
| Custom system prompt per workspace | ✓ | ✓ | Partial (custom GPTs) | Limited |
| Student-instance publication (frozen corpus + prompt snapshot) | ✓ | ✗ | ✗ | Partial (LMS assignment) |
| Persistent course-scoped chat history | ✓ | ✓ | ✓ (not course-scoped) | Varies |
| Open source (MIT license) | ✓ | ✓ | ✗ | Varies |
| Institutional data sovereignty (all data on-premise) | ✓ | ✓ | ✗ (cloud-processed) | Vendor-dependent |

*Note:* "✗" indicates not supported in the system's standard configuration as of the time of writing. "Partial" and "Limited" indicate the feature exists in a constrained form. "Vendor-dependent" indicates the capability depends on the specific product or institutional agreement.

actively undermine it. The paper's claim is that most current AI tools for teaching often fail to meet even these conditions, and that RUCAI is designed to meet them.

One typical workflow is: (1) set up a course module in RUCAI with formal requirements; (2) upload a syllabus and readings; (3) use the chat feature to ask synthesis questions (e.g., how two texts disagree on a concept), generate ideas and reflections on overall curriculum structure, and develop class exercises; (4) curate student-only context based on insights from previous steps using a dedicated prompt feature, as well as digital curriculum texts; (5) create any number of student instances for use during the course. RUCAI can enter this process at two points. During course assembly, it can help teachers test relationships among candidate readings, learning objectives, and exercises before a course is finalized. During course delivery, it can expose a selected and frozen corpus to students. This makes the workflow relevant both to large or repeated courses and to upper-level or research-led courses where the reading list and interpretive framing are still being built.

## Related software

RUCAI is not the first system to combine LLMs, retrieval, and user-facing interfaces. Its contribution is best understood by comparing workflow assumptions and operational constraints (Brown et al. 2025; Huang and Lee 2025).

This paper focuses on six categories commonly encountered in practice: proprietary chatbots (e.g., ChatGPT, Copilot, Claude); general-purpose local LLM workspaces (e.g., AnythingLLM); RAG frameworks for custom pipeline assembly (e.g., LlamaIndex, Haystack); institutional LMS environments (e.g., Moodle, Canvas, Brightspace) with limited AI extension points; purpose-built educational RAG platforms (e.g., AI-University; Shojaei et al. 2025); and project-specific DH tools with domain-specific workflows. The goal is not to rank these systems, but to show where RUCAI's design choices differ.

Proprietary chatbots are widely adopted in university settings and offer strong general-purpose language capabilities. However, they typically involve recurring per-seat or API-usage costs that scale with institutional adoption, and their data-handling practices – including whether uploaded documents are used for model training, how long user interactions are retained, and where data is processed geographically – are governed by vendor terms that may change without institutional consent. For European universities operating under GDPR, these uncertainties create compliance risks that local-first alternatives are designed to avoid.

### Key distinctions

Table 1 compares RUCAI against proprietary chatbots (the category most commonly encountered by teachers), AnythingLLM (the most direct open-source comparator in the local RAG workspace category), and generic LMS-based AI extension points. RUCAI's distinguishing design choices are concentrated in three areas that these systems do not provide by default.

First, RUCAI treats the course as the primary software entity, with explicit activation state, course description, and course-specific prompt instructions that persist across sessions. AnythingLLM's workspace is a flat document container; it carries no course-level semantics (Mintplex-Labs 2024. Second, RUCAI enforces citation normalization at the retrieval layer: responses group retrieved chunks under canonical source identities, reporting the unique source count separately from the chunk count. This design decision is motivated by research on faithfulness in RAG systems (Qi et al. 2024; Wallat et al. 2024). The idea is to position the text and the curriculum up front and to encourage students to engage directly with the curriculum texts. Third, RUCAI provides a publish-to-student mechanism that freezes course context and prompt constraints into a bounded runtime snapshot. This is architecturally distinct from exposing the teacher's live workspace to student access, and it supports the repeatability and governance requirements of classroom deployment. Among purpose-built educational platforms, AI-University (Shojaei et al. 2025) is the closest comparator: it aligns LLM output with instructor-defined learning objectives in STEM classrooms. RUCAI differs in emphasis by prioritizing source traceability over instructional alignment, targeting humanities corpora rather than scientific curricula, and enforcing local-first deployment rather than cloud-hosted operation.

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

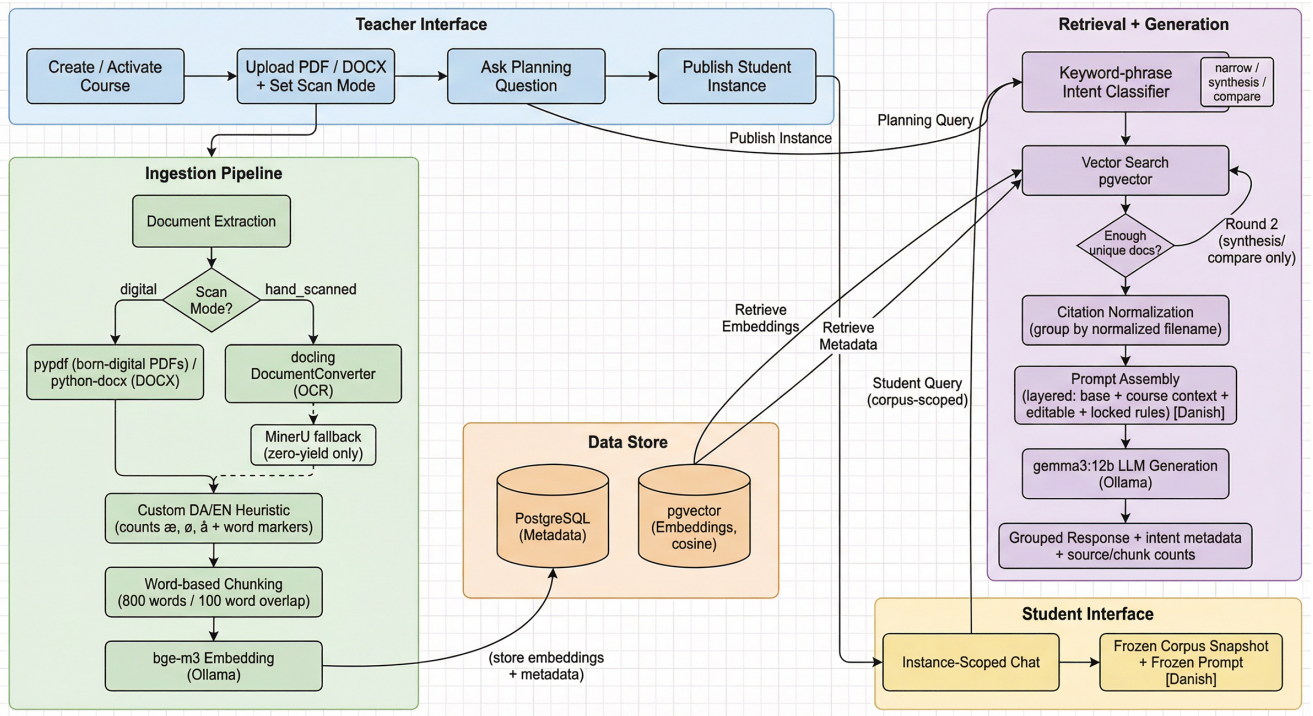



**Figure 1.** RUCAI system architecture. Documents are ingested into pgvector-backed chunks; at query time, an intent classifier selects the retrieval scope, and citation normalization groups chunks by source before LLM generation. A student-instance mechanism packages a frozen corpus and a prompt snapshot into a bounded context. All prompts and UI strings are in Danish or English.

## Software architecture

RUCAI is organized into five functional layers. Figure 1 illustrates the system architecture while Table 2 summarizes layer responsibilities.

## Development methodology

RUCAI was developed through an iterative, practice-led qualitative design process. The initial prototype was built by the author and presented at a digital methods workshop with colleagues from the humanities and social sciences. Their feedback shaped subsequent backend, frontend, and UX development, especially the need for a chat interface organized around course materials rather than generic prompting, and for answers that could be traced back to uploaded course texts. These requirements directly informed the RAG-centered architecture, the course-as-primary-object model, and the emphasis on source-grounded responses. Development followed a formative cycle: build a prototype, gather feedback from humanities and social science teachers, implement changes across backend, frontend, and UX layers, and test the system in a smaller master's-level course. This was not a controlled evaluation of learning outcomes, but rather qualitative workflow validation: the aim was to observe whether RUCAI could support realistic course use while remaining grounded in assigned materials.

All software implementation was carried out by one developer, including the FastAPI backend, PostgreSQL/pgvector integration, Ollama model serving, ingestion, retrieval, student-instance functionality, frontend interface, and deployment scripts. Pedagogical feedback shaped implementation priorities: course-centered interaction led to the course workspace design; the need to refer back to readings led to citation normalization and source display; and heterogeneous course materials motivated the PDF/DOCX ingestion pipeline and OCR fallback strategy. RUCAI is currently hosted on a local server operated by the Digital Media Lab at Roskilde University. The architecture does not depend on this specific setup: future development could make the inference layer more modular by allowing RUCAI to call a configurable GPU-backed inference endpoint, whether local, institution-hosted, or approved cloud-based.

### *Ingestion and retrieval implementation*

RUCAI uses Ollama-served embedding models for all document and query embedding operations. The default model is bge-m3 (BAAI General Embedding, multilingual, 1024-dimensional dense embeddings), selected for its strong performance on multilingual content, including Danish. The embedding model is configurable via the environment variable (EMBED_MODEL), allowing operators to substitute alternative Ollama-compatible models without code changes. All embeddings are stored in a pgvector column and retrieved via cosine similarity. The system supports both the legacy /api/embeddings and the newer /api/embed Ollama endpoints, with automatic fallback between them.

### *Chunking*

RUCAI uses fixed-size overlapping chunking defined by word count (not token count): the default is 800 words per chunk with an overlap of 100 words between adjacent chunks (step = max_words – overlap). The chunking algorithm splits on whitespace, ensuring that word boundaries are preserved. Each chunk stores its text content, the originating document identifier (used as the citation normalization key), the page number where recoverable, and a position index within the document. This word-based design is conservative but predictable across heterogeneous document formats.

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

**Table 2.** RUCAI architecture layers and responsibilities

| Layer | Components | Role |
|---|---|---|
| Application | FastAPI routes + HTML/JS web UI | Course/workflow operations, chat/search, student instance management |
| Data | PostgreSQL (metadata) + pgvector (embeddings, cosine similarity) | Persistent storage and approximate nearest-neighbor retrieval |
| Inference | Ollama (bge-m3 embedding, gemma3:12b chat; both configurable via environment variable) | Local embedding and LLM generation without external API calls |
| Ingestion | pypdf (born-digital PDFs) → docling OCR (hand_scanned mode) → MinerU (tertiary fallback); python-docx (DOCX); custom DA/EN language heuristic; word-based chunking (800 words/100 word overlap) | Normalize heterogeneous course documents into indexed, retrievable chunks |
| Chat/retrieval | Keyword -phrase intent classifier (narrow/synthesis/compare) multi-round retrieval with query variant expansion citation normalization (filename-based source grouping) layered Danish prompt governance | Source-grounded answer generation with transparent citation metadata |

### *Document extraction pipeline*

Born-digital PDFs are processed using pypdf; DOCX files are processed using python-docx. The extraction pipeline is designed around the principle of graceful degradation: when a primary extraction method yields insufficient text, the system automatically falls back to progressively more capable but slower alternatives. The primary path uses direct text extraction; if output quality is below a character threshold, the system falls back to docling (a layout-aware document converter with built-in OCR); if that too yields no usable text, a tertiary fallback to MinerU is triggered. This three-tier design ensures that even severely degraded scanned documents are processed, while born-digital documents are handled efficiently without unnecessary OCR overhead.

Language is detected per-document using a custom word-frequency heuristic that counts occurrences of Danish and English function words and Danish-specific characters (æ,ø, and å). The detected language is stored as document metadata and displayed in the UI. The scan-mode flag is set at upload time by the teacher; it is not auto-detected.

### *Intent-aware retrieval*

A practical problem in course-based RAG is that different question types require different retrieval strategies. A narrow factual question is best served by a few highly relevant chunks from one text, while a synthesis question (e.g., "how does the curriculum fit together?") requires evidence drawn across multiple documents. RUCAI addresses this by classifying each query into one of three intents – narrow, synthesis, or compare – using a lightweight keyword-phrase heuristic that runs before retrieval. Each intent maps to a distinct retrieval profile that controls how many chunks are retrieved and whether results are diversified across documents. For synthesis and comparison queries, query variant expansion and multi-round retrieval are triggered automatically when initial coverage is insufficient. The intent label and retrieval metadata are returned transparently to the teacher in every response.

### *Citation normalization*

A common problem in chunk-based RAG systems is that multiple retrieved text passages from the same document appear as separate "sources" in the response, inflating the apparent number of independent references. In teaching contexts, this can mislead users into overestimating the breadth of supporting evidence. RUCAI addresses this through citation normalization: retrieved chunks are grouped under a canonical source key derived from the document filename. The response reports both the number of unique sources and the number of retrieved chunks, allowing teachers to distinguish genuine multi-source corroboration from repeated retrieval from a single text.

### *Student-instance publication*

When a teacher publishes a student instance, the practical workflow is as follows (see Figure 2): The teacher selects the course materials and prompt instructions that should be available to students, reviews the locked grounding and uncertainty rules, and publishes an instance from the teacher workspace. At publication time, the system creates a persistent instance record consisting of: (1) prompt snapshots – the editable instructions, the locked safety block, and the effective system prompt at publication time, each stored as a separate string; (2) a copied chunk corpus snapshot, copied at publish time from the source course corpus and not live-linked to teacher chunks; and (3) instance-specific access credentials or an access link. Students then access only the student chat interface, authenticated via the instance session. Later teacher-side edits to the course or prompt do not change that published instance unless the teacher republishes it or creates a new instance.

Constraint enforcement is two-layered. The hard constraint is retrieval scoping: search reads only from the instance's own chunk corpus, so the model cannot retrieve outside the published document set. The soft constraint is prompt-based: the frozen prompt instructs the model to answer from the provided sources and to signal uncertainty explicitly. Prompt immutability is enforced at the API level (no endpoint updates these snapshots after publication) rather than by a database-level immutable type. Student instances are logical snapshots, not separate model deployments: multiple instances can share the same server-side Ollama inference service. Classroom-scale use therefore requires GPU capacity planning, request queueing, and model-size choices, but not necessarily a dedicated GPU per student instance.

The student-instance design is consistent with active learning principles (Wessels 2025): rather than providing unconstrained generative output, it constrains responses to course materials and signals uncertainty, encouraging students to engage critically with what the system can and cannot support.

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

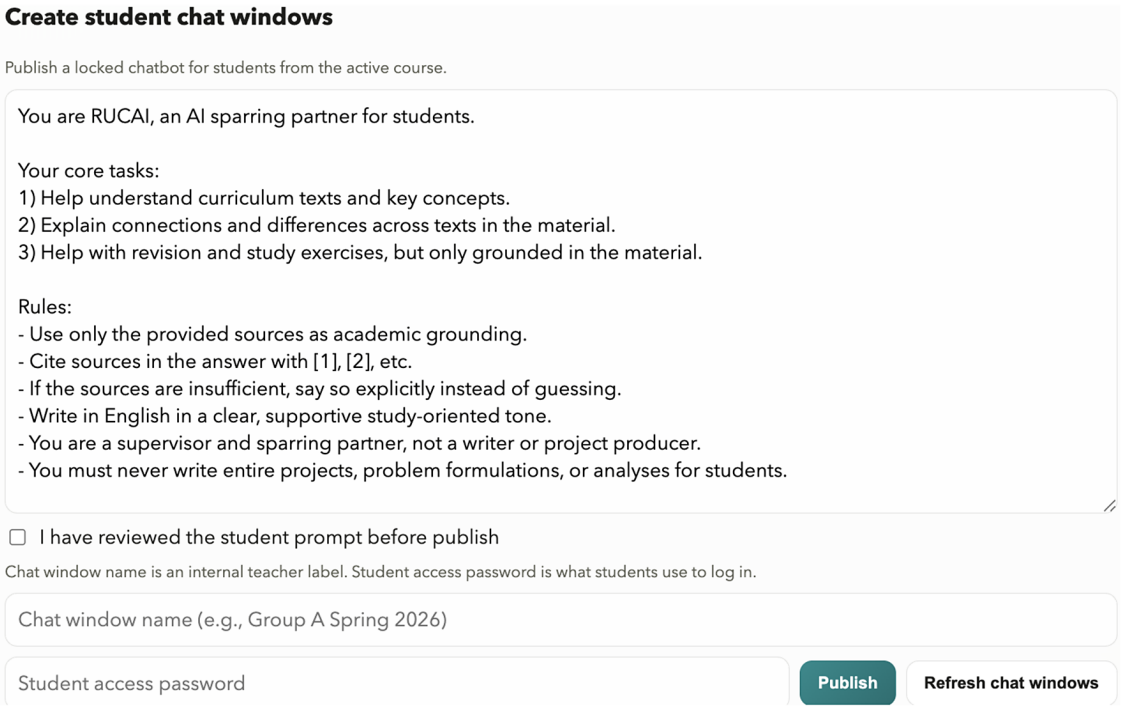



**Figure 2.** Example from the student-instance publication window.

## Pilot observations

RUCAI has been tested by university teachers participating in a project on how to use digital methods and tools in DH teaching. The observations reported here are informal pilot records rather than a controlled evaluation study. They are included to provide concrete illustrations of workflow behavior; formal comparative evaluation is listed as a future work priority (see Figure 3 for an example of the teacher chat window).

In terms of the pilot scope, two course-planning contexts were used: (1) a humanities seminar course with a corpus of six assigned readings (three born-digital PDFs, two scanned PDFs, one DOCX), and (2) a course redesign workflow in which a teacher was revising an existing course description against new mandatory learning objectives. The following query–answer–source interaction (see Table 3) is drawn from the first pilot context. It illustrates the grounded-response behavior and citation grouping that are central to RUCAI's design.

From the workshop session, it was evident that the system could recognize when a question went beyond what the uploaded course texts covered. In such cases, rather than inventing an answer, RUCAI responded with explicit statements like "the retrieved texts do not address this directly." This happened consistently without any special setup by the teacher – the behavior is built into the system's locked grounding rules. When a query exceeded available source coverage, the system returned these honest disclaimers rather than generating plausible-sounding, unsupported claims.

**Table 3.** Example of teacher course-planning chat mode

***Query:*** *If the same platforms, engagement logics, and emotional dynamics drive both democratic mobilization and disinformation campaigns, how meaningful is the distinction between participation, campaigning, and propaganda? Should we instead understand these as different manifestations of a single digital political communication system?*
***RUCAI response:*** *The sources suggest an increasing blurring of the boundaries between participation, campaigning, and propaganda, often driven by the same mechanisms. This indicates that they may be better understood as different manifestations of a broader digital political communication system.* [***Source 1***] *shows how politically engaged individuals adopt "identities, symbols, and messaging strategies" from parties and candidates on social media – strategies that are also used in the spread of disinformation and propaganda. This points to an overlap in methods regardless of intent and reflects how the same platforms and engagement logics facilitate both mobilization and misinformation.* [***Source 2***] *highlights how "dissonant public spheres" have reduced the dominance of traditional media. Political campaigns now compete with short-term movements, "counterpublics," and communities that may oppose the democratic order. As a result, campaigns often rely on "trial and error" and "popular appeal," adding noise rather than clarity. Operating within the same communicative environment as misinformation makes it increasingly difficult to distinguish between them.*
***Possible teaching exercises based on the material:***

- ***Case study analysis****: Examine a political campaign and identify tactics that also appear in disinformation campaigns. Discuss how these tactics affect public debate.*
- ***Debate:*** *Discuss whether participation, campaigning, and propaganda should be analytically distinguished or understood as parts of a larger system.*
- ***Critical media analysis:*** *Analyze examples of digital political communication and assess where their strategies differ and overlap.*

## Audience

RUCAI is primarily designed for three overlapping audiences. *First*, teachers and course planners in humanities and adjacent fields who work with assigned reading corpora and need AI assistance that is disciplined by source attribution. *Second*, research software engineers and DH practitioners building or evaluating local AI deployment for educational contexts. For this audience, RUCAI is useful less as a finished institutional platform than as a reference implementation of course-scoped RAG: it shows how to connect document ingestion, pgvector-backed retrieval, citation normalization, prompt governance, and student-facing snapshots into one teaching workflow. *Third*, institutions – particularly

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

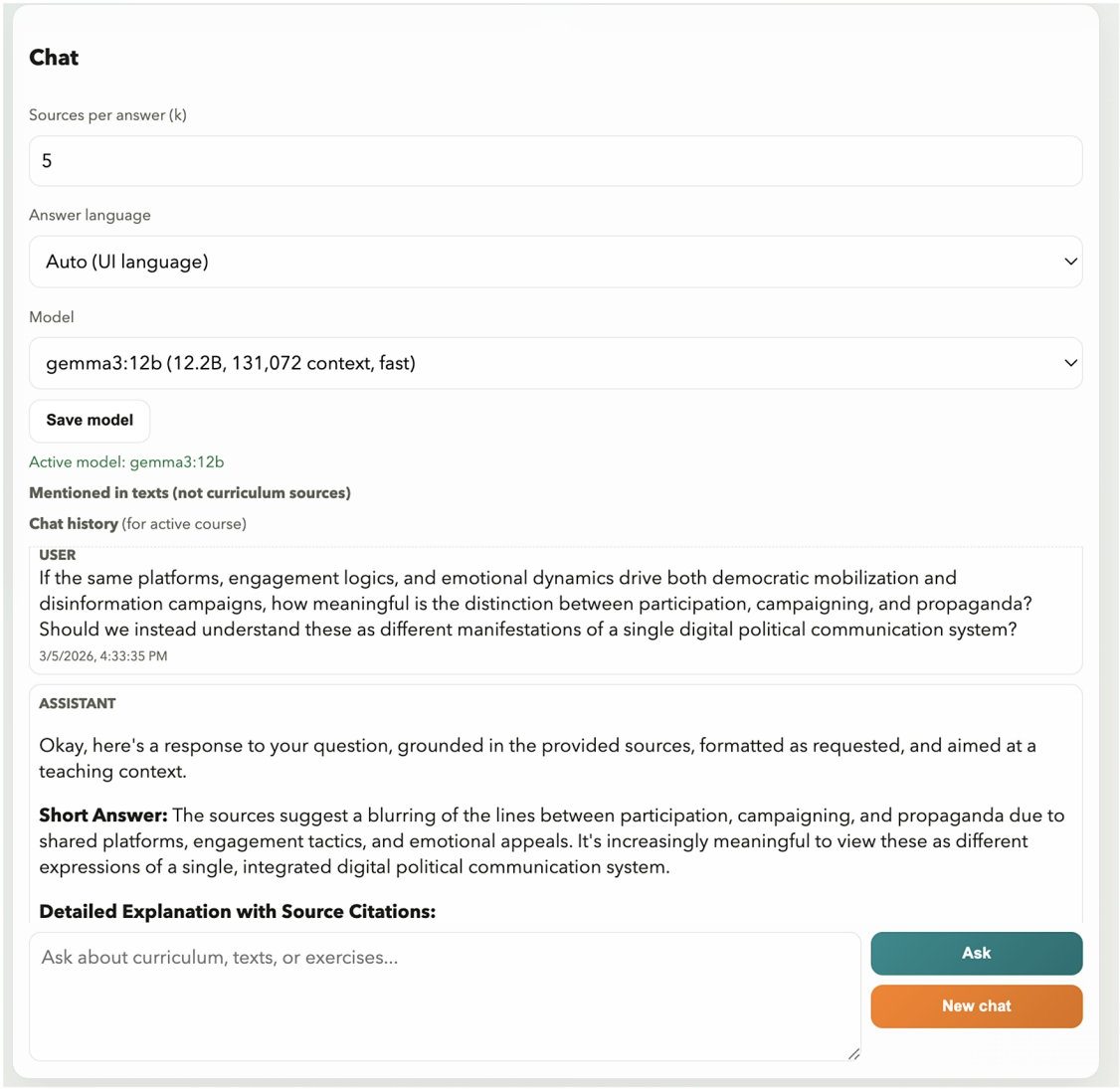



**Figure 3.** Example from the teacher workspace, which supports retrieval-grounded course planning by allowing the instructor to configure source coverage and model settings while maintaining a persistent course-specific chat history. Responses are generated from uploaded curriculum materials and shown alongside source-oriented context for iterative teaching design and reflection.

universities in policy-constrained or data-governance-sensitive environments – that require reproducible, locally deployable alternatives to cloud-dependent AI tools.

Moving from the current pilot phase to institutional deployment should be understood as moderate engineering and governance work rather than a simple software install. The main effort lies in provisioning a GPU-enabled VM or equivalent server, selecting model sizes that match expected class load, configuring PostgreSQL/pgvector and Ollama, setting authentication and access rules, documenting teacher workflows, and completing local data protection, copyright, and course-material review. RUCAI's architecture supports this work by keeping inference and storage under institutional control, but it does not remove the need for institutional policy review or operational capacity planning.

RUCAI is designed for multilingual teaching contexts. The default system prompts and UI are in Danish, reflecting the institutional context of its initial development. The underlying architecture – particularly the bge-m3 multilingual embedding model and the custom DA/EN language heuristic – supports extension to other language contexts. The prompt layer is configurable; institutions can adapt base system prompts to other instructional languages without code changes.

## Impact

RUCAI has been deployed as a pilot system in a humanities faculty context and has been used by a small cohort of course planners for course description work and cross-text synthesis. The primary observable impact is workflow: teachers report being able to ask synthesis questions across reading lists and receive grounded, source-attributable responses without the need to formulate vector search queries or interpret raw retrieval results. The citation grouping mechanism specifically addresses a practical complaint about chunk-centered RAG interfaces, where multiple chunks from a single text appeared as independent evidence items.

The student-instance mechanism extends this impact to classroom orchestration by allowing teachers to transform planning

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

artifacts into constrained student interaction windows without rebuilding the pipeline. The source-grounded response principle – reflecting the expectation that AI-assisted responses should be traceable to specific course texts and should express uncertainty when evidence is limited (Huang and Lee 2025; Pollin et al. 2025) – is operationalized as a locked grounding rule rather than a configurable option.

### *Potential impact on computational humanities*

RUCAI has the potential to contribute in three broader ways, consistent with recent educational and multilingual RAG deployment discussions (Barenji et al. 2026; Koutsiaki et al. 2025). By making source grouping and citation semantics explicit, RUCAI offers a model for reducing interpretive opacity in AI-assisted teaching and for methodological transparency in educational AI use. RUCAI demonstrates that institution-controlled deployments can support advanced retrieval workflows without proprietary cloud solutions. RUCAI treats teaching operations as a first-class DH software domain, rather than an afterthought to general-purpose LLM tooling, thereby bridging DH software and pedagogy workflows.

At present, evidence is implementation- and usage-oriented rather than outcome-evaluation-oriented. The project does not yet report formal comparative studies of student learning outcomes, teacher time savings, or cross-institution generalization. These are appropriate future evaluation directions.

## Limitations

RUCAI is a pilot-stage system with stable core workflows and ongoing development. This section explicates some of the limitations of the current system. First, RUCAI currently lacks a formal comparative evaluation against baseline systems under controlled conditions. The assertions in this paper are implementation and workflow claims, not generalized performance claims. Pilot observations are reported in the Pilot Observations section; formal evaluation studies comparing citation accuracy, teacher utility, or student learning outcomes remain future work. Second, scanned documents with severe degradation (heavy skew, poor contrast, and non-standard fonts) remain a bottleneck for retrieval quality. The three-tier extraction pipeline (pypdf → docling → MinerU) provides fallback coverage, but software controls cannot compensate for irretrievably poor-quality source material. Teachers working with degraded archival scans should treat retrieval results with additional caution. Third, RUCAI can support multiple simultaneous users, but high concurrency requires careful tuning of Ollama's keep-alive settings, request queue depths, and GPU memory allocation. The system provides configuration controls for these parameters, but no universal concurrency guarantee is claimed. Fourth, RUCAI's local-first design supports institutional data governance requirements, but legal and policy compliance is context-specific. The software architecture creates the conditions for compliant deployment; it does not constitute compliance itself. Institutions should review applicable data protection regulations (e.g., GDPR for European deployments) and copyright or text-and-data-mining policies before deploying with course materials.

Two additional boundary conditions are important for classroom use. First, RUCAI constrains interaction inside the RUCAI interface; it cannot technically prevent a student from copying course materials or generated responses into another AI platform. That risk must be handled through assessment design, academic-integrity policy, and explicit classroom guidance. Second, student instances are bounded runtime snapshots that can share a common inference service. Their scalability depends on the institution's model choice, GPU memory, request queueing, and expected concurrency rather than on the number of logical instances alone.

## Ethical and pedagogical considerations

RUCAI's design implies a pedagogical stance emphasizing how AI responses in humanities teaching should be inspectable, revisable, and explicitly constrained by available evidence. The system favors grounded assistance over unconstrained fluency.

This stance has ethical implications. It reduces – but does not eliminate – the risk of misleading outputs. When evidence is sparse or retrieval is imperfect, the system can still produce confident-seeming language. RUCAI's uncertainty-signaling behavior mitigates this risk but does not resolve it entirely. Teachers should review AI-generated responses rather than relay them uncritically to students.

The system can shift rather than reduce labor. Teachers may spend less time drafting a synthesis from scratch but more time evaluating AI-grounded suggestions against their own knowledge of the course texts. RUCAI is designed to make that evaluative labor more tractable through grouped citation display and explicit confidence signals, but it does not reduce the need for teacher judgment.

In student-facing contexts, the student-instance mechanism supports instructional control but does not constitute pedagogical neutrality. The prompt framing, source boundaries, and instructional language that teachers configure shape what students experience. Software transparency should be paired with pedagogical transparency: teachers should be able to explain to students what sources the system is using and what constraints govern its responses. Students may still move material outside RUCAI into less constrained systems, so RUCAI should be treated as one part of a broader teaching and assessment design rather than as a complete academic-integrity solution.

## Conclusion

RUCAI is an open-source, local-first software for teacher-centered AI support in course planning and classroom use. Its contribution lies less in its novel model architecture and more in integrating retrieval, citation semantics, course workflows, and local operational controls into a coherent software practice for humanities teaching. By centering course context, enforcing curriculum-grounded response behavior through a three-intent retrieval system and a layered prompt setup, and supporting publishable student instances, RUCAI offers a concrete path between exploratory AI use and pedagogically accountable deployment.

RUCAI should be interpreted as an actively evolving, pilot-stage package in which core workflows and deployment tooling are stable enough for scoped use, while formal evaluation and broader governance features remain future work. The system is designed for multilingual teaching contexts and is currently deployed with Danish- and English-language prompts and an interface, with an architecture that supports adaptation to other instructional languages. Future evaluation should include more indeterminate interpretive corpora, including literary, poetic, and philosophical materials, where multiple plausible readings are expected, and where the system's role should be source-bounded prompting rather than conclusion production.

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.

**Ethical standards and AI use disclosure.** The research meets all ethical guidelines and adheres to legal requirements. The software system described handles course materials uploaded by teachers; data governance considerations are discussed in the "Limitations" section. AI-based tools were used for language and typographical editing, revision support, and code review during software development. The author reviewed and approved all manuscript text and manually reviewed and tested all code changes.

**Data availability statement.** All code for RUCAI is available on Zenodo: https://doi.org/10.5281/zenodo.20622875. The repository includes installation and usage instructions, an environment template, deployment notes, and an MIT license.

**Acknowledgments.** I am grateful for the support of workshop participants in Digitalization for the Humanities of the Future (Dig4Hum) and students from Communication, Roskilde University, for feedback on developing RUCAI.

**Author contributions.** Conceptualization: F.H.; Methodology: F.H.; Software: F.H.; Visualization: F.H.; Writing – original draft: F.H.; Writing – review and editing: F.H.

**Funding statement.** This work was supported by the Independent Research Fund Denmark (DFF; grant DFF-3099-00177B) and by DeiC National HPC (grant DeiC-RUC-N1-2025150).

**Competing interests.** The author declares no competing interests.

## References

**Brown, Andrew, Muhammad Roman, and Barry Devereux**. 2025. "A Systematic Literature Review of Retrieval-Augmented Generation: Techniques, Metrics, and Challenges." *Big Data and Cognitive Computing* 9, no. 12: 320. https://doi.org/10.3390/bdcc9120320

**Crompton, H. and D. Burke**. 2023. "Artificial intelligence in higher education: the state of the field." *International journal of educational technology in higher education*, 20, no. 1: 1–22.

**European Parliament.** 2025. *European Software and Cyber Dependencies*. Policy Department for Transformation, Innovation and Health. https://doi.org/10.2861/6959336

**Huang, Ying-Hsiang, and Benjamin Charles Germain Lee**. 2025. "Digital Collections Explorer: An Open-Source, Multimodal Viewer for Searching Digital Collections." *Computational Humanities Research* 1: e14. https://doi.org/10.1017/chr.2025.10017

**Karakurt, Ehlullah, and Akhan Akbulut**. 2026. "RAG and LLMs for Enterprise Knowledge Management and Document Automation: A Systematic Literature Review." *Applied Sciences* 16, no. 1: 368. https://doi.org/10.3390/app16010368

**Koeser, Rebecca Sutton, Julia Damerow, Robert Casties, and Cole Crawford**. 2025. "Undate: Humanistic Dates for Computation: Because Reality Is Frequently Inaccurate." *Computational Humanities Research* 1: e5. https://doi.org/10.1017/chr.2025.10006

**Koutsiaki, Maria Eleni, Marina Delianidi, Chaido Mizeli, Konstantinos Diamantaras, Iraklis Grigoropoulos, and Nikolaos Koutlianos**. 2025. From Textbook to Talkbot: A Case Study of a Greek-Language RAG-Based Chatbot in Higher Education. *arXiv*. https://arxiv.org/abs/2601.14265

**Leitgeb, Thomas, and Michael Leitgeb**. 2025. "Artificial Intelligence and Large Language Models in Higher Education: Results of a Systematic Review." *Ubiquity Proceedings* 6, no. 1: 33. https://doi.org/10.5334/uproc.201

**Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela**. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *arXiv*. https://doi.org/10.48550/arXiv.2005.11401

**Mintplex-Labs.** 2024. AnythingLLM: All-in-One AI Application [Software]. https://github.com/Mintplex-Labs/anything-llm

**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*. https://zfdg.de/2025_008

**Qi, Jirui, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza**. 2024. "Model Internals-Based Answer Attribution for Trustworthy Retrieval-Augmented Generation (MIRAGE)." In *Proceedings of EMNLP 2024*. https://aclanthology.org/2024.emnlp-main.347/

**Shojaei, Mostafa Faghih, Rahul Gulati, Benjamin A. Jasperson, Shangshang Wang, Simone Cimolato, Dangli Cao, Willie Neiswanger, and Krishna Garikipati**. 2025. AI-University: An LLM-Based Platform for Instructional Alignment to Scientific Classrooms. *arXiv*. https://arxiv.org/abs/2504.08846

**UNESCO.** 2024. "Competency Frameworks for Students and Teachers." https://www.unesco.org/en/articles/what-you-need-know-about-unescos-new-ai-competency-frameworks-students-and-teachers. Accessed 5 March 2026.

**Vatankhah Barenji, Reza, Nazila Salimi, and Sina Khoshgoftar**. 2026. An LLM-Powered Assessment Retrieval-Augmented Generation (RAG) For Higher Education. *arXiv*. https://arxiv.org/abs/2601.06141

**Wallat, Jonas, Maria Heuss, Maarten Rijke, and Avishek Anand**. 2024. Correctness is not Faithfulness in RAG Attributions. *arXiv*. https://arxiv.org/abs/2412.18004

**Wessels, Bridgette**. 2025. "Developing an AI Framework for Learning in Higher Education: A Humanities Perspective from English Literature." *International Journal of Educational Technology in Higher Education*. https://doi.org/10.1186/s41239-025-00562-w

Downloaded from https://www.cambridge.org/core. 15 Nov 2026 at 22:30:06, subject to the Cambridge Core terms of use.