heyprof – Eine KI-gestützte Plattform für die Lehre

Volker Reichenberger
NXT Nachhaltigkeit und Technologie
Hochschule Reutlingen

Dirk Schieborn
NXT Nachhaltigkeit und Technologie
Hochschule Reutlingen

5. März 2026, Version 1.0

heyprof ist eine webbasierte Plattform zur Lehrorganisation und Betreuung von Lehrveranstaltungen, die sich durch die konsequente Integration von Large Language Models (LLM) auszeichnet. Sie wurde zunächst für MINT-Vorlesungen an Hochschulen entwickelt, ist aber für alle Arten von Lehrveranstaltungen geeignet. Neben klassischen Funktionen wie Materialverwaltung, Aufgaben und Klausuren steht ein dialogbasierter KI-Tutor im Mittelpunkt, der Studierende durch den sokratischen Dialog führt, anstatt fertige Lösungen zu liefern. Dabei wir der Kontext und aktuelle Stand der Lehrveranstaltung berücksichtigt, um gezielte, niveaugerechte Unterstützung zu bieten. Dieses Dokument beschreibt die technische Architektur, die wesentlichen Funktionen, die zugrundeliegenden pädagogischen Prinzipien und einen Ausblick auf die Weiterentwicklung der Plattform.

1 Einleitung

Große Sprachmodelle können nutzbringend in der Lehre eingesetzt werden, häufig werden Lernende aber mit allgemeinen KI-Assistenten allein gelassen. Neben dem Problem des *cognitive bypass*[1] ergibt sich dabei häufig das Problem, dass die KI-Lösung nicht zum aktuellen Kenntnisstand der Studierenden passt. Die KI kennt nicht den aktuellen Stand der Lehrveranstaltung und liefert daher oft Antworten, die für Studierende zu komplex sind oder durch Verwendung nicht eingeführter Konzepte den Lernprozess behindern. Ein typisches Beispiel sind Lösungen zu Programmieraufgaben, die Sprachmittel verwenden, die nicht Teil der Lehrveranstaltung sind (z. B. Listenkomprehensionen, Generatorausdrücke, Typhinweise).

heyprof adressiert diese Anforderungen, indem die KI kontextsensitiv und gezielt unterstützt, Der KI-Tutor kennt den genauen Text der aktuellen Aufgabe, die Musterlösung sowie die privaten Tutorhinweise des Dozenten, den bisherigen Python-Code des Studierenden und relevante Passagen aus dem Kursskript. Er gibt zunächst *keine*

[1] Eine technische Hilfe übernimmt die kognitive Arbeit vollständig, was den Lerneffekt eliminiert.

vollständigen Lösungen, sondern führt durch gezielte Fragen und Hinweise, die exakt auf das Kenntnisniveau des Kurses abgestimmt sind.

Neben dem KI-Tutor deckt heyprof den vollständigen Lebenszyklus einer Lehrveranstaltung ab: Vorlesungsfolien und -aufnahmen, Übungsaufgaben mit automatischer Bewertung, Klausuren mit handschriftlicher Einreichung, Lernfortschrittsverfolgung und mehrsprachige Benutzeroberfläche.

2 Technische Architektur

heyprof basiert auf einem modernen Web-Stack, der für schnelle Iteration und minimalen Betriebsaufwand ausgelegt ist.

Frontend
: Next.js 15 mit React und TypeScript (App Router). Benutzeroberfläche mit Tailwind CSS und Radix-UI-Primitives. Mathematik wird clientseitig durch KaTeX und MathJax gerendert.

Backend
: Next.js API Routes (serverless) für alle KI- und Geschäftslogik. Supabase (PostgreSQL mit Row-Level Security) für Datenpersistenz und Authentifizierung.

KI-Dienste
: OpenAI `gpt-4o-mini` [1] für Chat, Bewertung, Übersetzung und Zusammenfassung; `text-embedding-3-small` für semantische Dokumentensuche (RAG [2]); Whisper v2 [3] für Vorlesungstranskription; DALL-E 3 für automatische Aufgaben-Cover-Bilder.

Speicher
: Supabase Storage für PDFs, Bilder und Audiodateien. Semantische Einbettungen werden in der `pgvector`-Erweiterung von PostgreSQL gespeichert.

Die Anwendung wird auf Vercel betrieben und benötigt keine eigene Serverinfrastruktur. Rollenbasierte Zugriffskontrolle (Dozent / Studierende) wird auf Datenbankebene durch Supabase RLS und serverseitige Prüfungen durchgesetzt.[2]

Das **Datenmodell** besteht aus folgenden Kernentitäten: *Course*, *Session*, *Material*, *Exercise* (mit Typ open-ended / multiple-choice / coding), *ExerciseCollection*, *Exam*, *Submission* und *MaterialChunk* (semantisch indexierter Textabschnitt).

3 Funktionen für Dozentinnen und Dozenten

3.1 Kurs- und Sessionverwaltung

Dozenten legen Kurse im Dashboard an und erhalten einen achtstelligen Zugangscode, mit dem sich Studierende einschreiben. Kurse lassen sich veröffentlichen oder sperren.

[2] Neue Nutzer werden in einer Whitelist-Tabelle freigeschaltet und erhalten dabei ihre Rolle (Dozent oder Studierende/r) zugewiesen.

Jeder Kurs ist in *Sessions* gegliedert, die einzelnen Vorlesungsterminen entsprechen und chronologisch angezeigt werden. Innerhalb einer Session lässt sich die Reihenfolge aller Elemente per Drag-and-Drop anpassen.

3.2 Materialverwaltung und Annotation

Hochgeladene PDF-Folien können per Freihand-Zeichenwerkzeug (Farbe, Linienbreite, Radierer) auf einzelnen Seiten annotiert werden. Studierende sehen diese Anmerkungen als Overlay über den Folien.

Nach dem Upload werden Materialien *ingested*:[3] Der Text wird extrahiert, in Abschnitte aufgeteilt, eingebettet und im Vektorspeicher abgelegt, sodass der KI-Tutor gezielt relevante Stellen aus dem Kursskript abrufen kann.

3.3 Aufgabeneditor

Der Aufgabeneditor unterstützt drei Typen: *Freitext*, *Multiple-Choice* und *Code* (Python). Für jede Aufgabe kann der Dozent hinterlegen:

- eine **Musterlösung** (auf Wunsch für Studierende sichtbar),
- **Tutor-Instruktionen** (privat, nur vom KI-Tutor verwendet),
- **Labels** und **Collections** zur Organisation.

Aufgaben lassen sich aus LaTeX-Quelldateien importieren, die dem `\begin{aufgabe}` / `\begin{loesung}`-Format folgen – dies ermöglicht die nahtlose Übernahme bestehender Aufgabensammlungen. Cover-Bilder werden automatisch per DALL-E 3 generiert.

3.4 Klausurverwaltung

Klausuren bestehen aus numerierten Fragen mit Punktzahl und einem Countdown-Timer. Fragen lassen sich aus gescannten PDF-Klausurblättern importieren: Eine KI-Pipeline extrahiert Texte, Nummerierung und Punkte automatisch. Nach der Klausur stehen alle Einreichungen mit KI-generierten Bewertungen zur Verfügung.[4]

3.5 Aufnahmen und Transkription

Ein browserbasierter Audiorekorder ermöglicht das direkte Aufzeichnen von Vorlesungen in heyprof; alternativ lassen sich fertige Audiodateien hochladen. Whisper v2 transkribiert die Aufnahme im Hintergrund. GPT-4o erzeugt anschließend eine gegliederte Zusammenfassung in mehreren Schritten: Chunking des Transkripts, Teilzusammenfassungen, Gesamtsynthese. Transkript und Zusammenfassung sind dann für Studierende sichtbar.

---

[3] *Ingestion-Pipeline:* Text-Extraktion seitenweise, Chunking auf ≤ 1500 Zeichen, Einbettung mit `text-embedding-3-small`, Speicherung im Vektorspeicher.

[4] Jeder Studierende sieht seine eigene Bewertung mit Feedback; der Dozent sieht eine Übersicht aller Einreichungen.

3.6 KI-Konfiguration

Auf Kursebene lassen sich Tutor-Instruktionen für Material- und Aufgabenfragen hinterlegen sowie der Tutor ganz deaktivieren. Die *Persönlichkeit* des Tutors (im System als „Soul"bezeichnet) ist durch Markdown-Dateien konfigurierbar – eine Soul beschreibt Tonfall, Unterrichtstil und Schwerpunkte des Tutors. Neben sachlichen und aufmunternden Souls gibt es auch volksnahe Varianten (Soul Proll: „x geteilt durch Null und dann …? Merkste selber, Digger, oder?").

4 Funktionen für Studierende

4.1 Dashboard und Lernstudio

Das Dashboard zeigt eingeschriebene Kurse mit Fortschrittsanzeige und Benachrichtigungen über neu freigegebene Materialien. Das *Lernstudio* listet alle Sessions in chronologischer Reihenfolge. Innerhalb einer Session stehen PDF-Viewer mit Annotations-Overlay, Vorlesungsaufnahme mit Transkript und Zusammenfassung sowie die Aufgaben des Termins auf einer einzigen Seite bereit.

Im PDF-Viewer können Studierende einen Bereich auf einer Folie markieren; der KI-Tutor erhält dann diesen Ausschnitt als Fokus-Hinweis und erklärt gezielt den markierten Inhalt.

4.2 Aufgaben

Für jede Aufgabe stehen mehrere Einreichungskanäle zur Verfügung:

1. **Text/Code** – direkte Eingabe mit KI-Sofortbewertung.
2. **Zeichenfläche** – Freihandlösung für mathematische oder diagrammatische Antworten.
3. **QR-Foto** – der Desktop zeigt einen QR-Code; das Smartphone scannt ihn, nimmt ein Foto auf und die Antwort erscheint automatisch auf dem Desktop.
4. **KI-Tutor** – sofortige sokratische Unterstützung im integrierten Chat, der den Aufgabentext und den aktuellen Code des Studierenden kennt.
5. **Variante üben** – die KI erzeugt eine strukturgleiche Aufgabe mit anderen Zahlen oder Funktionsnamen.
6. **Lösung anzeigen** – nach einem echten Lösungsversuch.

Der Python-Editor für Coding-Aufgaben basiert auf CodeMirror 6 und *Pyodide* (CPython als WebAssembly [4]).[5] Er bietet Syntax-Highlighting, Auto-Einrückung, einen *Ausführen*-Button und ein Live-Ausgabefeld. Die Ausführung ist in einer WebAssembly-Sandbox isoliert, sodass kein sicherheitsrelevanter Code auf den Server gelangt.

[5] Der Python-Code läuft vollständig im Browser – keine Serverressourcen werden belastet. Ein 10-Sekunden-Timeout beendet Endlosschleifen. Gefährliche Importe (`subprocess`, `socket`, `ctypes`) werden clientseitig blockiert.

4.3 Klausuren

Die Klausur-Ansicht zeigt einen Countdown-Timer und einen Fragennavigator mit Fortschrittsanzeige. Antworten können als Text, Zeichnung oder QR-Foto eingereicht werden. Hinweise sind pro Frage in begrenzter Anzahl verfügbar. Nach Klausurschluss sieht der Studierende seine Punktzahl pro Frage mit KI-generiertem Feedback.

5 KI-Funktionen und pädagogische Integration

5.1 Der Vorteil gezielter KI-Nutzung

Der zentrale Unterschied zu allgemeinen KI-Assistenten liegt im Kontext. Bevor der KI-Tutor antwortet, wird sein System-Prompt mit folgenden Informationen angereichert:

- vollständiger Text der aktuellen Aufgabe,
- private Tutor-Instruktionen des Dozenten für diese Aufgabe,
- aktueller Python-Code des Studierenden,
- semantisch abgerufene Passagen aus den Kursmaterialien,
- handschriftliche Annotationen des Dozenten auf der aktuellen Folie,
- markierter Folienbereich (falls der Studierende einen Ausschnitt ausgewählt hat).

Dieser kursspezifische Kontext erlaubt dem Tutor, präzise und niveaugerechte Unterstützung zu geben, *ohne* die Antwort vorwegzunehmen. Eine allgemeine KI kennt weder das verwendete Lehrbuch, noch welche Python-Konstrukte im Kurs bisher eingeführt wurden – heyprof schon.

5.2 Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation bezeichnet eine Technik, bei der ein Sprachmodell vor der Antwortgenerierung gezielt mit relevantem Kontext aus einer externen Wissensbasis versorgt wird [2]. Das Modell selbst wird dabei nicht verändert; stattdessen wird der Prompt mit passenden Textpassagen angereichert.

Das Verfahren läuft in zwei Phasen ab:

Ingestion-Phase. Beim Hochladen eines Dokuments wird sein Text seitenweise extrahiert und in Abschnitte (*Chunks*) von maximal 1500 Zeichen aufgeteilt. Jeder Chunk wird durch das Einbettungsmodell `text-embedding-3-small` in einen numerischen Vektor (Embedding) überführt, der die semantische Bedeutung des Textes komprimiert kodiert. Diese Vektoren werden zusammen mit dem Originaltext in der `pgvector`-Erweiterung von PostgreSQL gespeichert.

Abfrage-Phase. Sobald ein Studierender eine Frage stellt, wird die Frage ebenfalls eingebettet. Dann werden die k Chunks gesucht, deren Embedding-Vektor dem der Frage am ähnlichsten ist (Kosinus-Ähnlichkeit). Diese Passagen werden in den System-Prompt des KI-Tutors eingefügt, direkt neben dem Aufgabentext und den Tutor-Instruktionen.

heyprof wendet RAG auf alle hochgeladenen Kursmaterialien an. Fragt ein Studierender *„Was versteht man unter einer verketteten Liste?"*, so sucht das System die relevantesten Abschnitte aus dem Kursskript – nicht aus dem allgemeinen Weltwissen des Modells –

und zwingt den Tutor, exakt mit den Konzepten und der Notation zu antworten, die in der Vorlesung eingeführt wurden. Das verhindert Antworten mit nicht eingeführten Sprachmitteln oder Konzepten und stellt sicher, dass der Tutor den Lehrstand des Kurses widerspiegelt.

5.3 Sokratischer Modus und Erklärmodus

Im **sokratischen Modus** stellt der Tutor ausschließlich Gegenfragen, die den Studierenden einen Schritt vorwärtsbringen, ohne die Lösung zu verraten. Ein typischer Ausschnitt:

> *Studierende/r:* Ich weiß nicht, wie ich mit der Rekursion anfangen soll.
> *Tutor:* Gute Frage. Überlege: Was ist der einfachste mögliche Eingabewert, für den du die Antwort sofort weißt, ohne rechnen zu müssen?

Im **Erklärmodus** erklärt der Tutor direkt, aktiv und anschaulich – beispielsweise wenn ein Studierender eine Folie nicht versteht. Er verwendet konkrete Beispiele, Alltagsmetaphern und nimmt ausdrücklich Bezug auf den bereitgestellten Folieninhalt.

5.4 Automatische Bewertung

Textantworten werden durch einen strukturierten GPT-4o-Aufruf bewertet, der ein JSON-Objekt mit Richtig-/Falsch-Urteil und kurzem Feedback zurückgibt. Handschriftliche oder fotografierte Lösungen werden durch das Vision-Modell analysiert, auf Relevanz geprüft und in LaTeX-Notation überführt. Die Bewertung aktualisiert den Fortschritt sofort.

5.5 Weitere KI-gestützte Dienste

Aufgabenvarianten
Die KI erzeugt strukturgleiche Varianten mit anderen Zahlen, Namen oder Funktionen – ideal für Wiederholungsübungen.

Aufgabenübersetzung
Auf Knopfdruck wird eine Aufgabe zwischen Deutsch und Englisch übersetzt (LaTeX-Formatierung und Code bleiben erhalten).

Klausurhinweise
Der Tutor gibt pro Frage begrenzte sokratische Hinweise auch während der Klausur – nie die Lösung, aber zielführende Impulse.

OCR handschriftlicher Annotationen
Handschrift auf Folien-Annotationen wird durch das Vision-Modell in Text bzw. LaTeX überführt und dem Tutor als Kontext bereitgestellt.

6 Pädagogische Prinzipien

Das Design von heyprof orientiert sich an bewährten Konzepten der Lernwissenschaft.

Konstruktive Abstimmung [5]**.** Jede Aufgabe, jede Tutor-Instruktion und jede Collection wird mit einem konkreten Lernziel verknüpft. Die KI nutzt die privaten Hinweise des Dozenten, um den Studierenden zur erwarteten Lösung zu führen – ohne sie preiszugeben.

Zone der nächsten Entwicklung [6]**.** Der sokratische Tutor versucht stets, die Grenze des aktuellen Verständnisses zu identifizieren und eine Frage zu stellen, die genau einen Schritt darüber hinausgeht.

Deliberate Practice [7]**.** Die Varianten-Funktion ermöglicht unmittelbares Üben nach dem Verstehen des Originals – selbes Konzept, neue Zahlen, sofortige Bewertung.

Multimodaler Input. Studierende unterscheiden sich in ihren bevorzugten Ausdrucksformen. heyprof akzeptiert Tipp-, Zeichen- und Fotoeingabe gleichermaßen und reduziert so die Diskrepanz zwischen Denken und Einreichungsformat.

7 Ausblick

heyprof wird kontinuierlich weiterentwickelt. Angesichts der Fülle der Möglichkeiten fällt es den Autoren schwer, eine verlässliche Prognose der folgenden Entwicklungsschritte zu erstellen. Hier ist eine unvollständige Auswahl.

Adaptive Lernpfade. Aktuell sehen alle Studierenden die gleichen Aufgaben in der gleichen Reihenfolge. Eine natürliche Erweiterung ist ein Empfehlungssystem, das auf Basis der individuellen Leistungshistorie die jeweils optimal herausfordernde Aufgabe vorschlägt. Aufgaben könnten dabei Voraussetzungsbeziehungen erhalten, die ein Kompetenzgraph modelliert.

Automatische Aufgabengenerierung. Ausgehend von einer Themenbeschreibung und einem Schwierigkeitsgrad könnte die KI neue Aufgaben – Text, Code oder Mathematik – vollautomatisch vorschlagen. Ein menschlicher Freigabeschritt bliebe erforderlich; der Autoraufwand würde jedoch deutlich sinken. Für Coding-Aufgaben würde ein automatischer Test-Runner die generierte Musterlösung vor der Freigabe verifizieren.

Interaktives Feedback zur Codeausführung. Der aktuelle Python-Editor zeigt rohe `stdout`-Ausgabe. Eine sinnvolle Erweiterung ist die Integration von automatischen Testfällen: Der Code des Studierenden wird gegen Eingabe-Ausgabe-Paare geprüft; der Tutor erhält die Testergebnisse als Kontext und kann sehr gezielt auf falsche Ausgaben hinweisen.

Kollaborative Aufgaben. Viele reale Programmieraufgaben sind kollaborativ. Ein Pair-Programming-Modus würde zwei Studierenden erlauben, an einem gemeinsamen Editor zu arbeiten. Der Tutor würde beide Beiträge sehen und zwischen ihnen vermitteln.

Lernanalysen. Aggregierte Daten zu Abgabezeiten, Fehlermuster und Hint-Nutzung würden Dozenten zeigen, welche Themen besonders viel Schwierigkeiten bereiten – als Grundlage für gezielte Kursüberarbeitungen.

Sprachinterface. Eine optionale Spracheingabe würde die Hürde senken, Fragen auf Deutsch zu formulieren – insbesondere für internationale Studierende.

Anbindung an Hochschulsysteme. Die Integration von LDAP/SAML ermöglicht Single-Sign-On und automatische Kurseinschreibung über die Systeme des Prüfungsamts.

Weitere MINT-Domänen. Der aktuelle Fokus liegt auf Programmierung und Mathematik. Erweiterungen für Schaltpläne, chemische Formeln, Musiknotation oder statistische Grafiken würden heyprof für weitere ingenieurwissenschaftliche Studiengänge öffnen.

8 Fazit

heyprof zeigt, dass KI-Unterstützung in der Hochschullehre nicht bedeuten muss, dass Studierende den Lernprozess umgehen. Indem der KI-Tutor tief in den Aufgaben-Workflow eingebettet wird, pädagogischen Kontext injiziert bekommt und auf sokratische Führung beschränkt bleibt, wird aus einem potenziell schädlichen Werkzeug ein wirksames Lernmittel.

Studierende erhalten rund um die Uhr sofortige, personalisierte, niveaugerechte Unterstützung – ohne den kognitiven Aufwand zu umgehen, der echtes Lernen erfordert. Dozenten gewinnen eine effiziente Autorenumgebung, automatische Bewertung und einen strukturierten Kanal für ihre Lehrmedien.

Die Plattform ist quelloffen und erfordert keine eigene Serverinfrastruktur. Die Veröffentlichung des Codes erfolgt, nachdem im Sommersemester 2026 die ersten Lehrveranstaltungen mit heyprof durchgeführt wurden und die dort gesammelten Erfahrungen in die Weiterentwicklung eingeflossen sind.

Nicht zuletzt ist heyprof selbst ein Produkt KI-gestützter Entwicklung: Ein Großteil des Codes und der Dokumentation entstand mit Hilfe von Claude [8] – ein praktischer Beleg dafür, dass KI-Assistenten, richtig eingesetzt, die Produktivität kleiner Entwicklungsteams erheblich steigern können.

Literatur

[1] OpenAI. *GPT-4 Technical Report*. Techn. Ber. arXiv:2303.08774. OpenAI, 2023.

[2] Patrick Lewis u. a. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: *Advances in Neural Information Processing Systems* (*NeurIPS*). Bd. 33. 2020, S. 9459–9474.

[3] Alec Radford u. a. "Robust Speech Recognition via Large-Scale Weak Supervision". In: *Proceedings of the 40th International Conference on Machine Learning* (*ICML*). 2023, S. 28492–28518.

[4] Andreas Haas u. a. "Bringing the Web up to Speed with WebAssembly". In: *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation* (*PLDI*). 2017, S. 185–200. DOI: 10.1145/3062341.3062363.

[5] John Biggs und Catherine Tang. *Teaching for Quality Learning at University*. 4. Aufl. Maidenhead: McGraw-Hill / Open University Press, 2011.

[6] Lev S. Vygotsky. *Mind in Society: The Development of Higher Psychological Processes*. Hrsg. von Michael Cole u. a. Cambridge, MA: Harvard University Press, 1978.

[7] K. Anders Ericsson, Ralf T. Krampe und Clemens Tesch-Römer. "The Role of Deliberate Practice in the Acquisition of Expert Performance". In: *Psychological Review* 100.3 (1993), S. 363–406. DOI: 10.1037/0033-295X.100.3.363.

[8] Anthropic. *The Claude 3 Model Family: Opus, Sonnet, Haiku*. Techn. Ber. Model Card. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. Anthropic, 2024.