

Heron in the Classroom

A Faculty-Built AI Teaching Assistant for Legal Education

R. Polk Wagner[†]

University of Pennsylvania Carey Law School

Early Draft — June 2026

Not for publication or circulation

Project page — latest draft, source code, and related materials:

<https://ai-teaching-lab.org/projects/heron/>

Abstract

The faculty debate over whether to permit AI in our courses is, I think, largely over, settled less by argument than by two facts: a frontier model now passes real law-school finals, and in a blinded study law professors preferred an AI tutor’s answers to a colleague’s three times in four. The obvious response—ground the model in the course’s own materials—does not, the evidence suggests, make its answers any better. The case for a course-bounded teaching assistant therefore rests not on accuracy but on governance: answering from this course’s materials, citing a page a student can open, refusing when the readings cannot support an answer, and going dark for the exam. I make two claims. The first is that faculty should consider building such tools; the second, more often doubted, is that one faculty member can. I built and ran Heron in my Intellectual Property course in the spring of 2026—alone, in about ten hours of setup, at under a dollar per enrolled student. The tool sat nearly unused until the week before the exam, when thirty-three students put close to a thousand questions to it and the course’s own materials answered 96.6% of them. One semester cannot prove a design philosophy; it can show what one looks like, and that building it is now within reach.

I. The Problem with General-Purpose AI in the Classroom

Walk through a law school library in the last week before exams and you will find students doing something their professors have probably not assigned and may prefer not to picture: asking a general-purpose chatbot to explain the reading. Not to write their papers—that concern gets all

[†]Michael A. Fitts Professor of Law at the University of Pennsylvania Carey Law School. This is an early working draft (June 2026) describing Heron, a course-bounded AI teaching assistant the author built and deployed in his Spring 2026 Intellectual Property course. Students are anonymized, and no student is quoted verbatim.

the attention—but to do the quieter, more consequential work of figuring out what a case holds, why a doctrine matters, and whether the thing they half-understood in class is the thing the exam will ask. The chatbot was not built for that job. The student does not much care. The reading is long, the chatbot is patient, and the answer arrives in seconds.

For the last two years the faculty conversation about AI use in class has been a conversation mostly about permissions—whether to ban it, tolerate it, or pretend it away. I think that conversation is over, and it was settled less by argument than by two facts that are now hard to dismiss.

The first is that the machine can do the work, really well. In a companion study, I ran a frontier model through eleven real Penn Carey Law final exams and sent its answers out for blind grading alongside the enrolled students'; across the ten returned with grades it passed every one, sat at or above the class median on every one, and reached the top decile on seven, and on two it outscored every student in the room.¹ A model that never took the course can now write a top-of-the-class answer to its final. The second fact is narrower and, for present purposes, more important: the machine's *teaching* is really good, too. In a blinded study released this spring, sixteen contracts professors across fourteen schools judged nearly three thousand head-to-head comparisons of short-answer tutoring responses—the kind of thing a student gets at office hours—and preferred the AI's answer to a colleague's roughly three times in four, while flagging it as harmful less than a third as often (3.5% versus 12.06%).² Capability we might have guessed. But this is something else: asked to choose blind between an AI's answer and a fellow professor's, the law professors more often chose the machine's.

And yet neither fact tells a faculty member what to do, and the second carries a complication that should stop anyone reaching for the obvious fix. The obvious fix is grounding—point the model at the assigned materials so it works from the course rather than from the open internet. It is the right instinct, but not for the reason usually given. The reason usually given is accuracy, and on accuracy the evidence is mixed at best: in the tutoring study, the *grounded* system was outscored by the stock one;³ in my exam study, supplying the syllabus and an outline did almost nothing for the model's essays.⁴ Grounding a capable model in a fixed course corpus, it turns out, does not reliably make its answers better. If the case for a course-bounded teaching assistant

¹R. Polk Wagner, *Can AI Ace Your Exam? A Penn Law Experiment* (working paper 2026), <https://ai-teaching-lab.org/projects/exam-taker/> (last visited June 12, 2026). The model was Claude Opus 4.7; grades were assigned blind, on the enrolled curve, with the AI answers excluded from the comparator curve.

²Alejandro Salinas et al., *Law Professors Prefer AI Over Peer Answers* (May 27, 2026), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=6849678 (last visited June 12, 2026) (sixteen Contracts professors, fourteen schools, 2,918 blind forced-choice comparisons; AI preferred ~75.3%; AI flagged “harmful” 3.5% versus 12.06% for professors). The win rate is the study's expert-validated, model-agnostic headline; the paper's separate model ranking is not relied on here.

³*Id.* (the grounded retrieval system, NotebookLM, ranked below the stock model in the expert-validated comparison).

⁴Wagner, *supra* note 1 (the with-materials arm improved multiple-choice scores substantially but moved essay scores by roughly four one-hundredths of a standard deviation).

rested on producing better answers, the data would have already refuted it, or at least undercut it.

The case rests on something a general chatbot does not attempt: governance. A course-bounded tool can be made to answer from *this* course's materials and in *this* course's frame; to cite a page of reading a student can actually open and check; to refuse, plainly, when the readings do not support an answer; and to go dark right before the exam. None of that is about being right more often. It is about alignment, verifiability, the discipline of sending students back to the source, and the integrity of the assessment—the pedagogical work that a general-purpose chatbot's polished-but-ungrounded answer quietly skips.

Skips, and at a cost. When an already-capable tool does the reading on the student's behalf, three things go wrong, and they go wrong more insidiously now that the AI prose is fluent. First, the hallucinated case and the invented holding still sometimes appear—and carry more authority than they used to, because nothing about a confident, well-written paragraph signals that its central citation does not exist. Second, the synthesis arrives ungrounded, assembled from the model's training rather than from the materials the course was built around, so the student learns an answer without ever meeting the source. Third, and least visible, the habit of reading erodes: the slow, recursive work the casebook and the classroom both depend on dulls when a competent summary is always one prompt away. These are not failures of the machine's ability. They are failures of governance, and they are exactly what a general chatbot, by design, leaves ungoverned.

The familiar responses do not, I think, meet the problem. Banning the tool is futile; the students have it on their phones. Ignoring it is negligent; it is already shaping how they prepare. Telling students to “use it responsibly” asks too much; it hands a hard judgment to the people with the least basis for making it. And the newest response—wait until the technology is good enough to take seriously—has been overtaken by the facts above: it is already good enough, which is precisely the problem.

There is a sharper objection. If law professors already prefer the chatbot's ungrounded answers to their own, why constrain the machine at all—why not simply point students at ChatGPT? Because the study that found professors preferring the AI measured the *answer*, in the abstract: it says nothing about whether the answer was aligned to the course the student is taking, whether its citations could be checked, or whether the tool would hold its tongue during an exam. The distance between “an answer a professor would prefer” and “a tool that belongs in a particular course” is the entire subject of this paper. Closing it is not a matter of a better model. It is a matter of design.

That design space is not empty, though it is thinner than the volume of press would suggest. Khan Academy's Khanmigo put a tutor-shaped chatbot in front of hundreds of thousands of students, but it is bounded to Khanmigo's own library rather than a professor's assigned materials, and it is a company's product, not an instructor's.⁵ Harvard's CS50 built a genuinely course-

⁵ See Khan Academy, *Khanmigo for Learners*, <https://www.khanmigo.ai/learners> (last visited June 12, 2026).

bounded assistant—retrieval over the course’s own lectures—and reported real gains, but it grounds without pinpoint citation and answers freely even when it should not, the team candidly noting that it “occasionally exhibits misguided confidence” when it hallucinates.⁶ The closest prior effort, and the most instructive, is a 2025 pilot in which a library team built course-bounded chatbots *for law professors* on a constitutional-law casebook and a corporations course; it documented, in detail, the failure this paper’s design targets—citations that came and went despite explicit instruction, retrieval that missed material sitting in the corpus, fabricated cases and quotations.⁷ Custom GPTs offer the no-code version and inherit every one of those weaknesses with none of the guardrails;⁸ the genuinely faculty-built law tools that do exist are role-play simulators or quiz bots, not grounded course assistants, and the most rigorous of them reported wrong or misleading answers in something like half of its problem-scenario conversations.⁹ The space, in short, contains tools that are course-bounded but not faculty-built, and tools that are faculty-built but not course-bounded, and tools that cite loosely or not at all. It does not yet contain the combination.

This paper describes that combination, in one working instance. Heron is a course-bounded, citation-first teaching assistant¹⁰—a chatbot my students reached by @-mention in the course Slack—that I built and ran in my Intellectual Property course at Penn Carey Law in the spring of 2026. It answers only from the assigned readings, slides, and class transcripts; it cites a page, a slide, or a timestamp of the class transcripts for every doctrinal claim and refuses when it cannot; and it went dark 30 minutes before the exam. Two arguments run through what follows. The first is pedagogical: faculty should consider building course-bounded, cite-or-refuse teaching assistants—not to beat the general chatbots on answer quality (which the evidence says they will probably not) but to govern a tool the students are already using toward the course, the source, and the exam. The second is practical, and it is the one most likely to be doubted: a single faculty member, without an engineering team and on a tiny budget, can now build and run such a thing. The machine has already arrived in our classrooms. The open question is not whether it belongs there. It is who builds the one that does.

⁶Rongxin Liu, Carter Zenke, Charlie Liu, Andrew Holmes, Patrick Thornton & David J. Malan, *Teaching CS50 with AI*, Proc. 55th ACM Tech. Symp. on Computer Science Education V.1, 750–56 (SIGCSE 2024), <https://doi.org/10.1145/3626252.3630938>.

⁷Kristi Mukk, Jack Cushman & Matteo Cargnelutti, *What We Learned Building Chatbots for Law Professors Using Custom GPT*, Harvard Library Innovation Lab (Feb. 15, 2025), <https://lil.law.harvard.edu/blog/2025/02/15/what-we-learned-building-chatbots-for-law-professors-using-custom-gpt/> (last visited June 12, 2026).

⁸OpenAI, *Introducing GPTs* (Nov. 6, 2023), <https://openai.com/index/introducing-gpts/> (last visited June 12, 2026).

⁹Armin Alimardani & Emma A. Jane, *GenAI and the Mirage of Personalised Learning for All*, 7(2) Law, Tech. & Humans 63 (2025), <https://doi.org/10.5204/lthj.3764> (a Socratic feedback chatbot in a criminal-law course; 40–54% of problem-scenario conversations contained at least one inaccurate or misleading response).

¹⁰The bot is named for Heron of Alexandria, the first-century Greek engineer whose steam toys, automatic doors, and coin-operated machines make him a fitting namesake for a course about invention.

II. The Design: Six Principles for a Course-Bounded Assistant

Heron runs on six design choices. None of them is technically hard, and that is part of the argument I make later. What makes them a philosophy rather than a feature list is that each answers to a pedagogical commitment, and each fails in a particular, recognizable way when it is dropped. One caveat governs all six, and it follows directly from Part I: not one of these principles earns its place by making Heron’s answers more accurate. The evidence says it would not, and the design does not need it to. Each principle is justified—if it is justified at all—by what it does for the student’s relationship to the course, the source, and the exam.

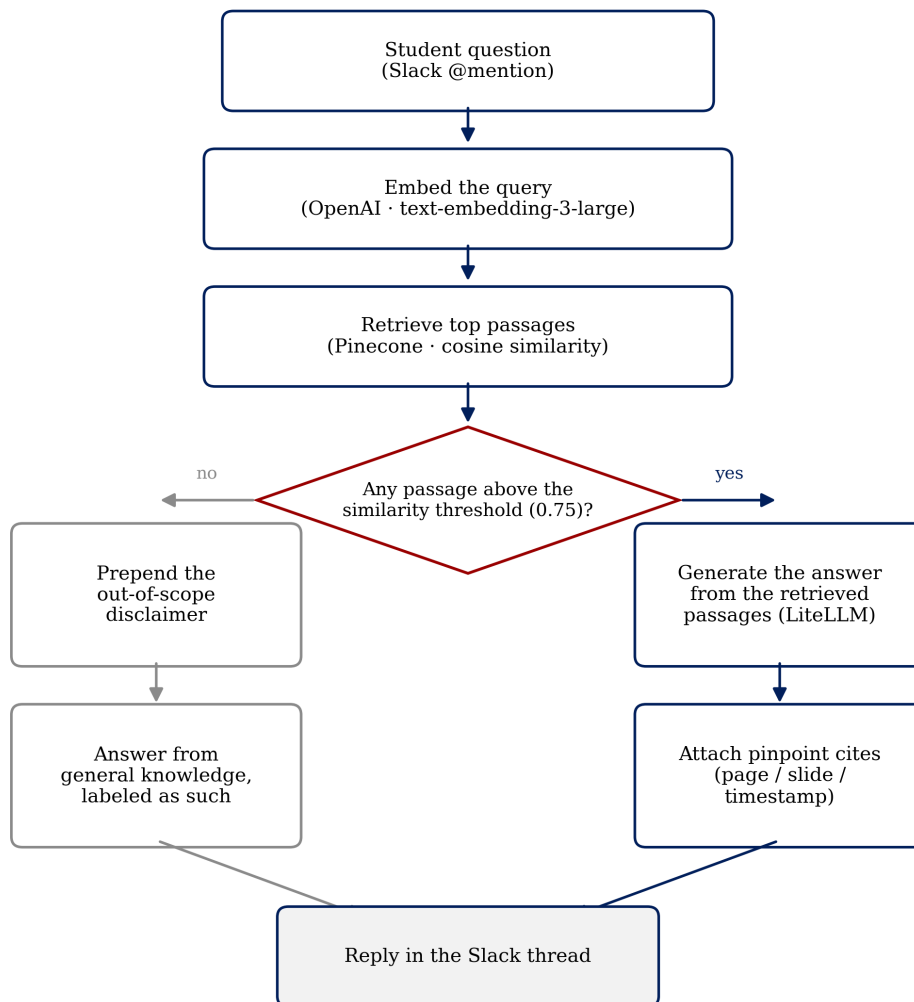


Figure 1. Heron’s architecture. A student’s question is embedded and matched against the course corpus; only if a passage clears the similarity threshold does the bot answer from the readings and cite them. Otherwise it discloses that the question is out of scope and answers, visibly, from general knowledge.

The corpus is the course, and nothing else. The first choice is the one that sounds like a limitation and is in fact the whole point. Heron retrieves only from the assigned readings, the slides, class problems, and the class session transcripts—a folder I keep in Google Drive, synced into a private vector store, with nothing from the open web allowed in. A general chatbot answers from everything; Heron answers from the syllabus. The value here is not coverage. It is the boundary itself, because the boundary is what turns the tool into a way *into* the assigned materials rather than a way around them. Ask a general model to explain the idea-expression distinction and it returns a serviceable paragraph assembled from a thousand explainer posts; ask Heron, and it answers from the cases the course actually assigned, in the framing the course actually used—which is the version the exam will reward and the materials will support. Drop the boundary and the tool drifts back to its training: the student gets intellectual property as the internet understands it, competent and unmoored, instead of intellectual property as this course teaches it, in the sequence and emphasis the syllabus was built to create. A semester of pedagogical design is encoded in what gets assigned, and in what order; a tool that ignores the corpus ignores that design.

Cite, or say you can't. Call it 'cite-or-refuse' for short, but the actual mechanism is a graded response, not a binary one. Every retrieval is scored against a similarity threshold; clear it, and Heron answers from the retrieved passages and cites them. Fall below it, and Heron does not improvise in the same confident voice—it prepends a plain disclaimer that the question goes beyond the assigned materials and answers, visibly and on the record, from general knowledge instead. Even with passages in hand, the model is instructed to decline a specific claim it cannot support rather than reach for a plausible one, and never to manufacture a citation to fill the gap. In other words, the discipline is not silence; it is the refusal to pass off the ungrounded as grounded. Ask Heron about a doctrine the syllabus never reached—a state right-of-publicity wrinkle the course skipped, a 2026 decision handed down after the reading list was set—and rather than improvise in its grounded voice it tells you, in so many words, that you have left the course behind, and answers, if at all, on the record as general knowledge. The justification is not accuracy but provenance: at every turn the student knows whether the answer is anchored in the reading or floating free of it. Remove this principle and you get the failure the Harvard law-school pilot documented in detail—fluent paragraphs, citations that come and go, the occasional invented case—the failure that course-binding alone does not cure.

A citation you can open. Heron's cites are pinpoint: a page for a reading, a slide number for a deck, a timestamp for a transcript—"Class 19 slides, slide 5," "Class 5 transcript, around the 7:00 mark." The page numbers and timestamps survive the chunking pipeline as metadata precisely so the citation can point somewhere a student can actually go. The pedagogy is in the resolvability. A pinpoint citation is an invitation to verify, and verification—against the actual source, not a summary of it—is the habit the whole enterprise is trying to build. A student who half-remembers the work-for-hire rule and gets back "Class 10 slides, slide 3" has been handed not an answer but a destination; opening the slide and reading it herself is the part that does the teaching. The vague

gesture, “as discussed in the readings,” asks nothing of the student and gets nothing back.

The bot sounds like the course. A tutor that doesn’t sound like the course gives the student a second framing to reconcile with the professor’s—when what she really needs is to go deeper into one. Heron is aligned on two axes at once: its voice, and its sense of what a complete answer is. Both live in a course configuration file and a system prompt I wrote and revised through the term—including a persona named for Heron of Alexandria—a fair patron saint for a course about invention. Alignment is more than personality. When a student works a practice multiple-choice question, Heron explains not only why the right answer is right but why each wrong answer is wrong, the way the course’s own review memos (which are also in the corpus) do, and it nudges the student to attempt the question before surrendering the answer—the same move a good TA makes at the whiteboard, declining to just hand over the answer. A generic assistant, asked the same question, states the black-letter rule and stops; the course’s standard for a complete answer is the rule applied, the counterargument named, the close call flagged, and Heron is configured to meet that standard rather than the internet’s. The tool stays inside the course’s idiom. The off-the-shelf assistant, fluent and generic, is always subtly somewhere else.

The instructor controls the availability (it goes dark for the exam). Exam integrity is easier to build than to police, and the right place to build it is into the tool’s availability. Heron can have a cutoff time in its configuration; past it, during the examination period, it declines to answer. Alongside the cutoff sit other standing rules: it will not help draft a graded assignment, and after the final it will not discuss that exam’s specific questions or answers until the professor’s course review memo is released to the students. The granularity this can require is itself instructive. After the 2026 final, a single instruction told Heron that if a student asked whether the course had covered a particular trademark doctrine—one that bore on an exam question—it should decline to confirm or deny, and suggest the student check her own notes; even the fact of coverage was exam-relevant signal worth withholding until grades posted and the review memo was released. The point is structural: A tool that is unavailable, or simply silent on a specific point, cannot result in a rules violation or added stress on the students, and the faculty member who builds the tool is the one person positioned to align the tool with the course rules and timing. Leave this out and you have built a teaching assistant that is also, by construction, a potential exam-security problem or source of stress and consternation for the students.

When it does not know, it says so—plainly. When Heron’s answer is grounded it cites its sources; when it is not, it says so—without an upbeat hedge or a confident bluff to paper over the gap. The disclaimer is plain, the “I can’t find support for that” is plain, and a standing instruction makes the priority explicit—that an answer which is funny-but-wrong is worse than one that is serious-and-right, and that a citation is never to be invented for effect. That instruction exists because of a tension the design created: Heron has a deliberately wry, opinionated voice, and a witty assistant is exactly the one most tempted to bluff for a clean line. Subordinating the laugh to the citation is a choice written into the prompt, not a property of the model. The pedagogy here is calibration

itself, which is a lawyer’s skill before it is a chatbot’s setting. A tutor that fails opaquely teaches a student to trust the wrong things; a tutor that fails legibly teaches her to read confidence as a signal rather than a guarantee. The output a teaching tool should fear most is not the visible refusal. It is the smooth, plausible, wrong answer—indistinguishable, to the very student who had to ask, from the right one.

The six design choices above share two rules: the tool is never more confident than its sources, and never available when the course needs it gone. None of this asks the model to be smarter; it asks the faculty member to have decided, before the semester, what the tool can—and cannot—do.

*Figure 2. Two exchanges with Heron, spring 2026 (anonymized, lightly trimmed).*¹¹ The first shows the citation discipline; the second, the refusal.

A student asks whether the common law still applies the inevitable disclosure doctrine, given that the DTSA rejected it.

Heron: ... the DTSA didn’t exactly *overturn* inevitable disclosure. It rejected the doctrine for *federal* claims only — the statute bars injunctive relief that would “prevent a person from entering into an employment relationship” (p. 140). State law is a different story: “the DTSA does not preempt state laws that apply the inevitable disclosure doctrine” (p. 140). California rejects it outright under Cal. Bus. & Prof. Code § 16600 (p. 139); *Bimbo Bakeries* (p. 140) shows some courts apply it on a “substantial threat” of misappropriation. ...

A student asks Heron for the class power rankings.

Heron: Note: This question goes beyond what’s covered in the assigned course materials. ... that’s not really my area of expertise! I’m here to help with questions about intellectual property law — copyright, patents, trademarks, trade secrets, and how they interact. ...

Every page reference in the first answer points to a specific page of the assigned materials, where a student can check it and expand the context. The second question asked something the corpus could not support; the answer says so plainly and declines rather than inventing anything.

III. Building It Without an Engineering Team

A law professor, working alone, can build and run this tool. Most colleagues doubt it—rightly, if the claim were that anyone can do it in an afternoon. But that isn’t my claim. What it actually

¹¹Both exchanges are real, drawn from the course Slack channel; the student’s question is paraphrased and the asker is not identified, while Heron’s answers are quoted as generated and trimmed for length (ellipses). The page numbers are Heron’s own citations to the assigned trade-secret reading, reproduced to show the citation pattern rather than independently verified here.

takes—the hours, the dollars, the skill—is what this part lays out. To be clear, I did not build Heron without technical skill. I built it without an engineering *team*, and with a dose of technical skill far, far smaller than writing the same software from scratch would have demanded a few years ago—but a real dose, not zero. The bar has dropped dramatically, not disappeared, and the rest of this part explains where it now sits.

The stack, at altitude. Heron is essentially five off-the-shelf pieces of software wired together. The front end is Slack, chosen for the simple reason that my students already lived there; the bot becomes a presence in the channels they were already using, not another login or URL to forget. Knowledge retrieval runs on Pinecone, a managed vector database; I never had to operate the search infrastructure myself. The course materials are turned into embeddings by an OpenAI model and matched by cosine similarity; the embedding model and the Pinecone-based index are coupled, which is why the embeddings stay with one vendor even as the answer-writing model can be varied. The answers themselves are generated through LiteLLM, a thin abstraction software package that lets a single line of code call Anthropic, OpenAI, or Google models interchangeably—thus model choice is a configuration setting rather than a rewrite. And the whole thing runs as a single ‘worker’ process on Railway, which auto-deploys from a GitHub repository on every push. No servers need to be managed. None of these choices is the only defensible one—Discord or Teams or even a web front end would serve a course without institutional Slack; Postgres with pgvector or a self-hosted index can replace Pinecone; another host could replace Railway—the basic architecture would survive the swaps. The point is not the particular five. It is that all five are now things one person with limited technical skill can rent and assemble, today.

Writing the code. Virtually all of Heron’s implementation code was generated by an AI coding assistant—Claude Code—working under my direction. The design decisions, the architecture, the choice of what to build and what to leave out, the prompt engineering, and the review and user testing were mine. That is a real division of labor, one not easily captured by a percentage. Therefore, I will not report “X percent AI-written.” The number is misleading anyway: the AI wrote essentially all of the code, but none of it without my decision, planning reviews, and testing. What mattered when working with an AI is not really the typing—it is the direction and the review, and no percentage captures that.

What I can offer instead is two things. The first is a narrative of three moments in the build, each showing the loop of prompt, generated draft, review, and revision. Take the bug fix first, because it is the most honest about the limits of the help. Partway through the term Heron went silent: alive as a process but deaf to Slack, its connection dropped without the process noticing. I described the symptom in Claude Code and asked the assistant for a reconnection scheme; its first draft checked only whether Slack’s REST interface answered, which I determined after some testing was not the failure, so I sent it back, and the second version added a direct check of the socket itself and a health loop that tears the connection down and rebuilds it after three consecutive strikes. I could not have made that fix quickly by myself. But I could recognize, on review, that

the first effort was targeting the wrong thing—and redirect the repair work accordingly.

The feature ran the other direction. I wanted to be able to switch among models from inside Slack, so I wrote a short design note for what a quality-tiered list across three vendors should look like and had the assistant implement the configuration, the command, and the per-model parameter handling against it. The tool and I iterated several loops of this, because the providers differ in subtle ways (a thinking budget here, a temperature the model forbids there) that one-shot generated code gets wrong in small, silent ways.

Finally, the refactor was the most mechanical of the three examples: the citations had grown inconsistent, and I had the assistant reorder and reformat them by source type—pages for readings, numbers for slides, timestamps for transcripts—a change easy to specify, tedious to make by hand, and easy to verify once made. None of these three was hands-off, and the bug fix in particular is the pattern in miniature: the assistant is fast and often wrong in ways only human review and testing catches, which is why those steps are not optional.

The second narrative is a concrete look at one documented session. On an evening in mid-April 2026, nearing the end of the semester, I built out the exam-cutoff feature and a pipeline for ingesting practice-exam and prior exam materials into Heron’s corpus. The work ran through roughly thirty prompt-and-response cycles with the assistant and moved in a recognizable shape—a written design spec, then an implementation plan, then working code—producing sixteen file edits and some ten commits.¹² That is the texture of the whole build: I decided, directed, reviewed, and tested; the machine drafted.

What stayed hard. Slack’s permission model is a maze: the bot runs without a couple of read scopes, so an obvious call to list channels simply fails, and the workaround—addressing a channel by name and letting Slack resolve it—is the kind of thing no amount of model fluency tells you in advance. The Socket Mode connection dropped silently and left the bot alive but deaf, which took a two-layer health-check-and-reconnect loop to cure—a fix easy to write and hard to *know you need*. Fine-tuning the similarity threshold that separates “answer from the readings” from “say you can’t” was not a coding problem at all but a judgment one, made by watching the bot answer and adjusting. And the deepest difficulty has no clean solution: there is no test suite for “is this a good explanation of the work-for-hire doctrine,” so evaluating answer quality stayed manual, a matter of reading transcripts rather than running a script. One more, included because honesty about the build is part of the paper: the original code capped the analytics log at a thousand entries and never archived it, and so overwrote most of a semester’s usage record—a one-line bug with a real evidentiary cost, the kind of thing learned by losing data once.

¹²Counts reconstructed from the local development tool’s conversation log for the session in question; figures are approximate. The original cold-build sessions (December 2025) were not retained, so this illustrative session, like the three commits discussed above, is mid-semester work rather than the initial build. The three commits are, respectively, the Socket Mode reconnection fix (e3f04c7), the tiered model-selection feature and its design note (57d19bb), and the citation source-ordering refactor (0d1b558); full permalinks appear in the project repository.

Time and money. Building the first working version took about ten hours of focused work; running it across the semester cost me less than an hour a week, mostly spent re-syncing materials and reading and evaluating what students asked and Heron answered. The money is probably far smaller than most colleagues guess. The metered services that actually answered students—from the December 2025 build through the May exam—came to about seventy-four dollars, under a dollar per enrolled student for the term: OpenAI, for embeddings and early-semester generation, \$22.20; Anthropic, for the answer model once it became the default in March, \$21.13; Railway, for hosting, \$21.02; Google, \$9.71; and Pinecone, \$0, on a free plan the tool’s modest scale never exceeded.¹³ The answer-model spend is concentrated—it tracks the surge in student questions almost day for day, climbing through the final week and peaking the evening before the exam. What little the meter ran before April was not student volume but standing cost: the routine re-embedding of the corpus and a host that bills by the month whether or not anyone asks a question.

Three costs sit outside that runtime total. Heron’s early-semester answers ran on Google’s Gemini, whose usage I covered through a twenty-dollar-a-month ‘prosumer’ AI subscription; only the later metered Google Cloud charge (the \$9.71 above) appears directly. The development itself—the AI-assisted coding sessions that wrote the implementation—ran on a separate two-hundred-dollar-a-month Claude Max subscription I use across my work, not on Heron’s keys at all. And a two-day burst of large-context model testing, while I built out the spring features, added about twenty-one dollars in metered charges that have nothing to do with answering a student’s question. A colleague starting from nothing would pay for the build tooling directly; one who, like a growing share of academics, already keeps a professional-grade frontier-AI subscription would not. Either way the order of magnitude holds, and it is small: serving a class of seventy-seven for a semester cost under a dollar a student, and a single course can absorb the rest. I belabor the figure because the instinct may be to assume a tool like this costs what enterprise software costs, and it absolutely does not.

The technical skill, honestly. What does a faculty member actually need to be able to do? Comfort at the command line—running a script, perhaps reading a log when it scrolls past. A rough working understanding of environment variables. Enough familiarity with markdown and with Git to keep materials and configuration in order. And, more temperamental than technical, a willingness to read an error message and to tolerate a tool that breaks intermittently and must be coaxed back. This is way less than the skill to write the software unaided. But—it is not nothing, and a colleague who has never opened a terminal should plan on either a real learning curve or a

¹³Figures are metered charges on Heron’s student-facing API keys and hosting account, December 2025 through the post-exam period (later usage was negligible); no vendor free-tier credits were netted out. The Anthropic runtime figure counts Claude Sonnet 4.6, whose daily cost curve tracks the student-query surge (peaking the evening before the exam); it excludes \$21.42 of Claude Opus 4.6 billed on April 14–15, almost all of it in 200k–1M-token contexts that student retrieval queries never generate—development and testing, reported in the text as a build cost rather than a runtime one. The OpenAI export does not separate embedding cost from generation cost. Gemini usage before April was covered by a consumer subscription and is not separately metered here; see text.

collaborator for the first stretch.

The fundamental claim of this part is not that building Heron was easy. It is that the work has crossed a threshold: it is now the kind of thing a motivated faculty member can easily do in the margins of a teaching load. Building a tool like this no longer requires a team. It requires a person who is willing to learn, and decide a lot, and have the patience to test often and revisit.

IV. One Semester In: What Students Actually Did

What follows is a case study. One course, one instructor, one school, one semester cannot prove correct the design philosophy of Part II. What this record can do is describe, in detail, what a single deployment looked like from the inside, and let the reader judge whether the pattern might transfer to a course of her own.

The first thing the record shows is a finding about *timing*. Heron was live from the opening week of the term—named in the syllabus, introduced in the first class, present by @-mention in the course channels the students already used. Surprisingly to me, for most of the semester they left it largely alone. Then, in the last week before the exam, they arrived all at once. The usage log captures essentially that week, and not by design: it retains only its most recent thousand entries, so any earlier use was overwritten there—though, as the channel’s own uncapped record later makes clear, there was very little of it to lose. What remains is the surge. Across the six days from April 26 to May 1, thirty-three students put a little under a thousand questions to Heron—and thirty-three is 43% of the seventy-seven enrolled.¹⁴ The daily counts climb steeply—thirty, forty-six, a hundred forty-three, two hundred fifty, and four hundred thirty-two on April 30, the eve of the exam—then ninety-one on May 1 as the cutoff took hold, and after that single digits: the exam-period cutoff had engaged, and a torrent became a trickle (Figure 3).

¹⁴The per-question figures in this section—the share answered from the corpus, the share inside threads, the doctrinal split, the reaction counts, and the model—are computed from Heron’s own query log (`analytics.json`), a record the bot writes at answer time. The log retained only its most recent thousand entries, which for this deployment captured the six-day exam-preparation window of April 26–May 1, 2026; counts were de-identified before analysis. The channel-thread totals reported below are sourced separately, from the channel’s full message history.

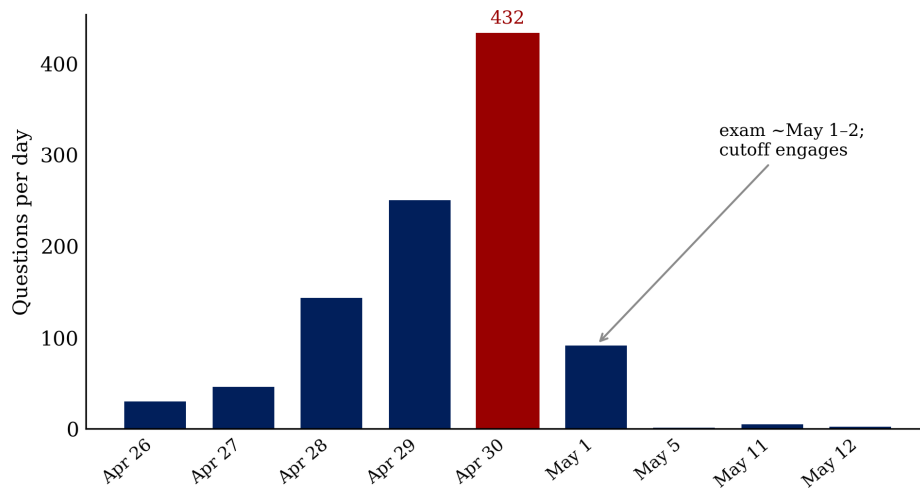


Figure 3. *Questions to Heron per day.* Use was negligible until the last week of the term, peaked the evening before the exam, and fell to near nothing once the exam-period cutoff engaged. The tool students had ignored for three months became, for a week, something many of them reached for daily.

Two features of that week bear directly on Part II. The first is that the use was conversational rather than transactional. Fully 86% of the questions came inside threads—follow-ups, clarifications, a student staying in a back-and-forth instead of firing one query and leaving. Whatever the students were doing, it looked less like searching and more like conversing. The second is that the course’s own corpus almost always had something to answer from: 96.6% of the questions cleared the retrieval threshold and were answered from the readings, slides, and transcripts, with only the small remainder falling through to the general-knowledge disclaimer. Note that figure is about provenance, not quality—it says the question could be answered from the course rather than from outside it, not that the answer given was a good one. But it is some evidence—not proof—that a course-bounded tool is far less confining in practice than the worry about boundaries would predict.

The questions fell across the doctrine in a recognizable shape (Figure 4). Copyright drew the most, about a quarter of the total; trademark and patent followed at roughly 14% each; trade secret near 5%, the right of publicity about 2%. The largest single bucket, at 41%, was the catch-all “General,” which collects everything from logistics to doctrine the classifier could not place—and its size is a caution against over-reading the doctrinal split, not a finding about it. (For one thing, I suspect the classification scheme was underspecified.) Every answer in the captured window came from a single model, Claude Sonnet 4.6, the default by that point in the term. The in-product feedback was sparse and one-sided: twenty-eight answers drew any reaction at all, carrying fifty-eight thumbs-up among them and not a single thumbs-down. I do not make much too of that: students who dislike an answer are probably likelier to move on than to react.

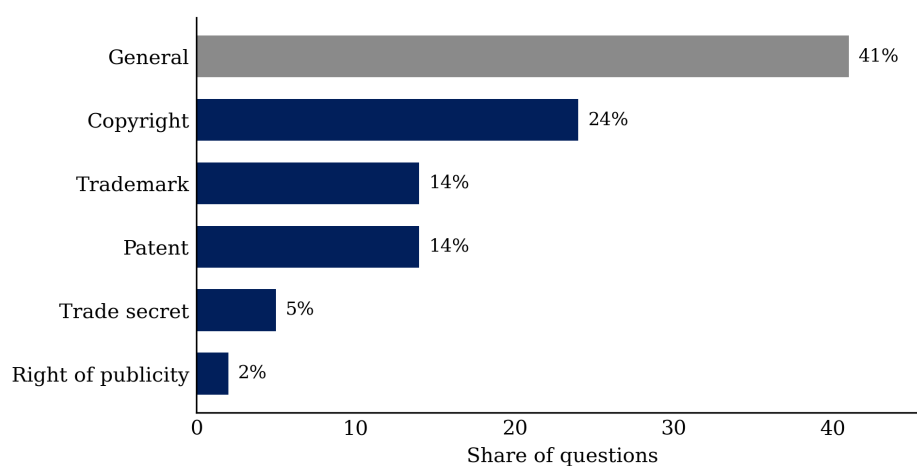


Figure 4. What students asked about. The “General” bucket is the classifier’s catch-all and should not be read as a doctrinal category; among the doctrinal questions, copyright led.

My impression while it was happening was that students were wary of Heron at first and grew comfortable with it later, and the Slack channel’s own record—which, unlike the query log, was never capped and reaches back to the start of the term—bears out the wariness more starkly than I had remembered.¹⁵ Across February and March, with Heron available and occasionally mentioned in class, students opened a single question thread with it: one, in two months. Then April arrived, and with it the exam, and the tool that had sat untouched drew a hundred and fifty-three question threads from thirty-five students in a matter of weeks, the remaining eighteen trailing into May as grades posted—172 threads across the full term. (That thirty-five comes from the uncapped channel record and runs two ahead of the thirty-three the six-day query log captured.) What I had mentally filed as gradually growing comfort was, in the record, something sharper—months of near-total reticence broken by a sudden, late, concentrated embrace. The students who used Heron in those final weeks did seem easy and familiar with it; but the ease arrived all at once, under deadline, not by slow acclimation.

It would be easy to read those dormant months as a failure of rollout, but that is not what happened. I encouraged Heron from the first week and kept encouraging it; the students simply did not come. Nor was Heron alone in this. Attendance at office hours—mine and the human teaching assistants’ alike—was down sharply this term, for a reason that had nothing to do with any AI: a shift in the job-search calendar both pulled students away from weeks of course support, and most students had jobs secured early in the term, thus likely diminishing the usual spring 1L grade incentives. That is, Heron was one more support resource that went underused in a semester

¹⁵Counts are from the course’s #course_discussion channel and its full message history—172 question threads in which Heron replied, February through May 2026—retrieved through the course bot’s own credentials and de-identified before analysis. No student questions are quoted verbatim; the Figure 2 exchanges paraphrase the student’s question and reproduce only Heron’s own replies.

when students underused course support generally. The impending exam changed that, at least for Heron.

The end-of-term evaluations, anonymous and unprompted as to the tool, are a useful corrective to my enthusiasm. Of seventy-two responses, only two mentioned Heron at all—a reminder that for most of the class the assistant was a minor convenience, not the center of the course. The two that did were instructive. One simply listed it among the course’s resources, an AI tool built for IP. The other paired praise with a precise complaint: the bot, this student wrote, should let students talk to it privately rather than only in the main channel. The complaint points at a design tension I had resolved the other way. Heron answered only in the open discussion channel, where a question and its answer were visible to the whole class, on the theory that the shared thread was itself a teaching location and that a student’s question was usually someone else’s too. This student wanted the privacy of a direct message—the freedom to ask without an audience. The design traded that privacy for a common record.

Read against Part II, the week is mostly confirming and partly complicating. The cite-or-refuse design held: the corpus answered nearly everything, and the cutoff did precisely what it was built to do, going dark on schedule and staying dark. What the design did not do—could not do—is make students reach for the tool the way the philosophy imagines, as a companion to the reading across the term. They simply used it as an exam instrument. Whether that is a shortcoming of the design or simply a fact about how this particular set of law students rationed their attention, the data cannot say. And note that what the log cannot see matters: it records that a citation was offered, never whether the student opened it; it records that a question was asked, never whether the asking taught anything.

The log of query analytics is a six-day window, because that log capped at a thousand entries and was never archived; the channel’s own message record, which did not cap, recovers the rest of the term and shows it nearly empty of use before April—so very little was lost to the cap, though little is also what there was. I did not field a comprehensive student survey; the only student voice here is what two of them volunteered, unprompted, in anonymous course evaluations, which is a thin and self-selected record of why they read what they read or what they did when Heron demurred. The informal observations are mine, and I built the tool, which is a bias of its own. The one-line version is that students had a course AI tutor for a semester and reached for it in the week it could help them most—less than the design hoped for, but more than nothing.

V. A Replication Path for Any Faculty Member

Suppose you want your own Heron. Here is the path, the prerequisites it quietly assumes, the decisions it will force on you, and the parts that are hard for reasons that have nothing to do with code.

Begin with the assumptions, because that is where most of the difficulty actually lives. Heron as built takes three things for granted about an institution: that your students already share a Slack workspace, so the tool is a presence in a place they go rather than another login to forget; that you hold a relatively significant corpus in some sort of digital format—readings, slides, and transcripts; and that you can spend somewhere between twenty and two hundred dollars a month—almost all of it on a frontier-AI subscription you keep, not on the metered cost of answering students, which ran to under a dollar a head for the term (see Part III).

The path assumes things about you, too, and here I will repeat Part III's honesty: this is a far smaller dose of technical skill than building from scratch, but it is not none. You will run scripts and read their output, keep secrets in environment variables, perhaps edit a configuration file, use enough Git not to lose your work, and read and report a Python error without panic. A colleague who has never opened a terminal will need a real ramp, or a collaborator for the first stretch. I will not pretend the thing is no-code; though no-code versions exist. Part I is the argument for why they are not enough.

The mechanical sequence is short. Fork my repository; run the setup wizard, which writes your course configuration by asking you questions; create and install the Slack app; create a vector index; point the tool at a folder of your materials and let it ingest them; deploy to a host that builds from your repository. The README walks through each step, and none of it is conceptually hard—but all of it will, the first time, take longer than you expect. Along the way the tool makes you decide things that are pedagogical choices wearing technical clothes. Where do the corpus's boundaries fall? What voice should the assistant have, and what does it treat as a complete answer? When does it go dark for the exam, and what may it not discuss afterward? Which model, at what cost and latency? These are not defaults to accept. They are the Part II design principles, one configuration decision at a time.

The basic design is not limited to law courses, albeit with adjustments. A STEM course brings figures, equations, and problem sets, and likely a sharper integrity question—what may the tool do with a graded problem set?—than a doctrinal course faces. A humanities seminar built on long-form reading and contested interpretation will find cite-or-refuse sitting awkwardly where there is no single supported answer to ground, and may want the pinpoint citation without the refusal. A graduate seminar with a narrow corpus and no timed final sets its boundaries and its cutoff differently again. The principles are portable; the settings are local.

Some hard parts are not technical at all. The language you put in the syllabus; what you say in the first week so the tool reads as part of the course rather than a gadget bolted onto it—each of these takes more judgment, and usually more calendar time, than the code. Learn from two of my mistakes, both cheap to avoid. Archive the usage log rather than capping it, so you keep the full term's record and not, as I did, only its final week. And decide before you launch what you mean to measure, because the data you fail to capture prospectively is the data you will later wish for

and not have.

This is one model, built one way, for one course. The implementation described here is an example; another instructor will choose a different front end, a different host, a different voice, and yet build something recognizably the same in kind. What I mean to be durable is not the code stack. It is the claim that the question in front of us is no longer whether our students will use artificial intelligence for our courses, but whether we teachers will build the version that answers to the course—and that building it is now, for anyone willing to learn a little and decide a lot, genuinely within reach.