

AI for Legal Help 2026 Class Report

Scoping, Building and Testing New Legal Aid Tech Systems

LAW/DESIGN 809E, Winter and Spring 2026

Margaret Hagan and Nóra Al Haider

Margaret Hagan ORCID: 0000-0003-2972-1867

Stanford Law School
Legal Design Lab

Legal Help Commons



Contents

THE CLASS

The course 5

The legal help challenges 6

The partnerships 8

The setup 10

Our class arc 13

FINDINGS AND TAKEAWAYS

Findings and Takeaways 18

Where does AI fit, where not? 18

Getting the scope & workflow right 21

How do we get high quality, safe solutions? 24

What partnerships, outreach, and collaboration are necessary to get a successful project? 30

How do we build with safety as a core goal? 34

Evaluation is hugely important now that building is so quick 37

AI Projects Need Intentional Sustainability Plans 42

NEXT STEPS

Next Steps: Coordination, Scale, Commons 45

The Legal Help Commons, legalhelpcommons.org



The class and teaching team, March 2026, Stanford Law School.

We share this report summarizing our activities, outputs, and insights from this past year's installment of our ongoing AI for Legal Help Class. It's a report for practitioners, teachers, researchers, funders, and builders working on AI for access to justice. Please let us know if you are working on similar topics (how to scale up legal help responsibly with tech?) or with similar methods (hands-on interdisciplinary group projects with amazing organizational partners)!

In our two-quarter class, we had 5 student teams partnered with 3 different legal aid teams, all working on different, highly scoped versions of the same challenge: whether and how AI could assist a legal team in scaling up a much-needed, hard-to-provide service to more people. Each student team, in close consultation with their legal team partner, went through a design, prototype, and evaluation cycle to see -- if between January and June -- they could get a solution that was effective, responsible, and closer to pilot.

As of Summer 2026, the tools are not ready for immediate, public-facing deployment. But we are working with partners and students to plan out how the current prototypes can move towards a controlled pilot within the legal teams. Aside from the pilot itself, each team developed a package of resources that can be immediately useful to anyone else building or testing AI tools in this space:

- **The functional agenda** of what a new technical system should do, to address a given legal team's challenge (like, determining a person's expungement eligibility; operating a housing hotline; or spotting defects in a 3-day eviction warning notice);
- **A working prototype** of a system that can perform these tasks, and that can be further built out;
- **The evaluation standards and materials** to judge a system's performance, to reliably tell if the system is safe, accurate, and sustainable enough to go to pilot;
- **The workflow roll-out plan and training** that would be needed to make this tool succeed in the given organization, with appropriate adoption and supervision.

We share this account of these projects' development, output, and status so that another teaching team, incubator, or legal aid organization can build off of these prototypes or take a similar approach to move the field further. We also share out what we did in 6 months, so that funders and builders can see what is realistic to expect in a university studio course. We made a great deal of progress, but also are honest about the data and technical work that are yet to be done -- and turn out to need a great deal more work to get a project pilot-ready.

The course

AI for Legal Help is a studio course that our Legal Design Lab team has taught yearly since 2023. It's cross-listed at Stanford Law School and the d.school, and attracts students from across many different departments on campus. In 2026, we ran the class as a 2-quarter sequence: Winter (January to March) as the initial scoping and prototyping of a new solution and Spring (April to June) as setting evaluation standards and refining the solution with this extensive testing. Before January 2026, our teaching team worked to find legal aid organizations to be the class partners and to scope specific legal workflows that students would attempt to transform with technology.

Students came from different backgrounds: law, computer science, data science, and policy-making, both undergrads and grad students. Most had never built an AI tool before, and most did not have familiarity with the legal topics that they would be working on. At the start of each quarter, they assembled into 5 different teams that were formed with a specific eye to having complementary skills and backgrounds. Each team was assigned to a specific challenge and legal team partner that they would work with throughout the class.

The class intentionally was set up to be as practical and grounded as possible, with higher-level learning objectives and insights coming from hands-on work on concrete legal team challenges. The goal was to help each student become a 'reflective practitioner' and to expand their own skills in disciplines outside of their own formal department. Each team worked closely with their legal team, immersed themselves in the minute details of the real workflow, and got regular feedback about how their technology, evaluation, and policy proposals would measure up in practice. The students had to make solutions that could work with real-world organizational constraints, funding and regulatory pressures, concerns about client data, professional duties, and people in crisis.

The legal help challenges

The student teams each took on 1 of 5 challenges that came from legal teams working on housing, eviction defense, and reentry services. For each challenge, the legal team had identified a current workflow that they hoped to transform. Could AI help legal teams offer this service in more robust, quality ways?











Housing Intake Hotline. The partner's housing telephone intake line runs only part of the week and cannot keep up with demand. Callers in the county call in, hoping to get help, but they wait on hold and many hang up before reaching an intake worker. The challenge was to figure out if AI could help with a new phone system that would handle the full intake for possible housing clients: eligibility check, conflict screening, and more than 100 questions, populating the case management system in real time and working in both English and Spanish.

Housing Accommodation Demand Letters. Attorneys at the partner team spend 30 to 60 minutes drafting a reasonable accommodation letter for tenants with disabilities. These letters are useful to request the landlord to change something about the rental home so the person's disability can be accommodated. Could AI help interview the client to get important details about their situation and wishes, and then generate a draft letter for attorney review? Could it do this for multiple accommodation types and in more than one language? Maybe also, could this be done via phone, text, or a web app?

Eviction Notice Defect Spotting. What is on the face of a 3-day, 30-day, or other warning notice about an impending eviction lawsuit matters a lot. If there are certain fields that are incorrect, missing, or incongruous -- this can mean the tenant has a defense to raise, and possibly get the eviction lawsuit dismissed. The challenge here was whether AI could help a volunteer or junior attorney spot these defects, when they look at a person's eviction warning notice. If a volunteer at a legal aid or court eviction clinic uploads a tenant's three-day eviction notice, can a tool effectively flag legal defects in it? Then, can it help the volunteer ask the tenant for other information to surface defenses and generate the required Eviction Answer (UD-105) to file?

Motion to Set Aside. When a tenant misses their eviction answer deadline or trial requirements, they often lose the lawsuit by default and then face set-out and lock-out, with law enforcement removing them and their belongings from the rental. But what happens when they didn't even know the lawsuit existed until the sheriff alerts them of this impending set-out? In this challenge, the question is whether AI could help a tenant reopen the lawsuit through a Motion to Set Aside (MSA) process. Could a tool guide a self-represented tenant through the MSA process, from intake through gathering documents, drafting a declaration through conversation with a chatbot, and assembling a court-ready packet that they can file and serve correctly -- so they get their day in court?

Expungement Screening. An estimated 1 in 4 Oklahomans has a criminal record, and a large share are eligible to clear some of the charges. But many people don't get their record cleared, even if it can be a big help for economic mobility and housing access. It can be complex to figure out, under the state regulations and case law, which charges are eligible to expunge & when. Could the team build a tool for junior attorneys or even expert practitioners to quickly determine which charges on a criminal record document are eligible for expungement? Could this system be traceable and rules-driven, so that it is a learning tool to teach the attorney the rules and support their professional development?

The partnerships

The class partnered with legal aid teams, who brought background details about the challenge they wanted help with, and a willingness to join the class regularly to meet with students, flesh out the specifics of what tech should and shouldn't do in their workflow, and give thorough feedback and examples to help the students build solutions that would be as likely to be impactful as possible. Most of our partners worked on the housing teams of their legal aid groups, though we had 1 team that works on reentry services for people coming out of the justice system.

Our teaching team had worked with each partner before the class to talk about the tasks where they thought tech could be impactful. What are the tasks that seem like they are so routine and templated that they could be handed off to an automated system? Where are there cases where demand from the public far outstrips legal teams' capacity to serve them, where tech might have a huge policy improvement? Where do you get excited about tech playing a role -- and where should tech not be brought in?

Partners joined on the first class of each term, to introduce their team, challenge, and goals to the students. They fielded questions as students tried to figure out the details of the legal situation and tech's role. Then throughout the quarter, attorneys and paralegals from the legal teams served as the student teams' clients and subject-matter experts. Staff at the partner orgs reviewed prototypes, corrected legal logic, and told teams when something would not work in practice. They also pulled in other officials, partners, clients, and colleagues to review the proposed tools and policies as well.

This partner-first structure is the backbone of the AI for Legal Help course. Our teaching team was there to find the legal aid teams, prepare them, and support both the students and the legal teams. But largely it was the frontline advocates who set out the design challenge, judged the students' performance, and pushed them towards more workable designs, workflows, and policies. The student teams did not get to decide on their own whether a tool was pilot-ready and likely to be impactful.

The setup

We held the class at a studio at the d.school. We also tried a new strategy for technical prototyping this year, in which we bought seats for each of the students on Replit, a platform that lets people talk in natural language to describe what web application they want to build, and then the platform works with them to code and implement the solution. Our hope was that this could augment students' abilities to quickly create working technical prototypes, in the quick prototyping-testing loops that we run in our design-driven classes. If Replit helped them execute working websites and tools more quickly, we hoped this could help the teams advance more quickly even if they didn't have extensive technical skills. Within the first weeks of Winter quarter, each team had a live, clickable prototype at a real URL, even if it was a rough draft that needed extensive content, usability, and safeguard improvements. This method of having live, interactive prototypes helped the students to test more regularly and critically with partners and users throughout the 2 quarters, rather than getting stuck in descriptions of possible solutions (which yield less insightful feedback, typically).

In past years' design studios, we did user and expert testing of the new innovations students had created. This year, we gave more structure to the testing rounds. In Spring quarter, especially, we had the student teams oriented around testing as the primary goal. They had to figure out if this prototype is ready to pilot in the field -- or if it is not, what it would take to get it pilot-ready?

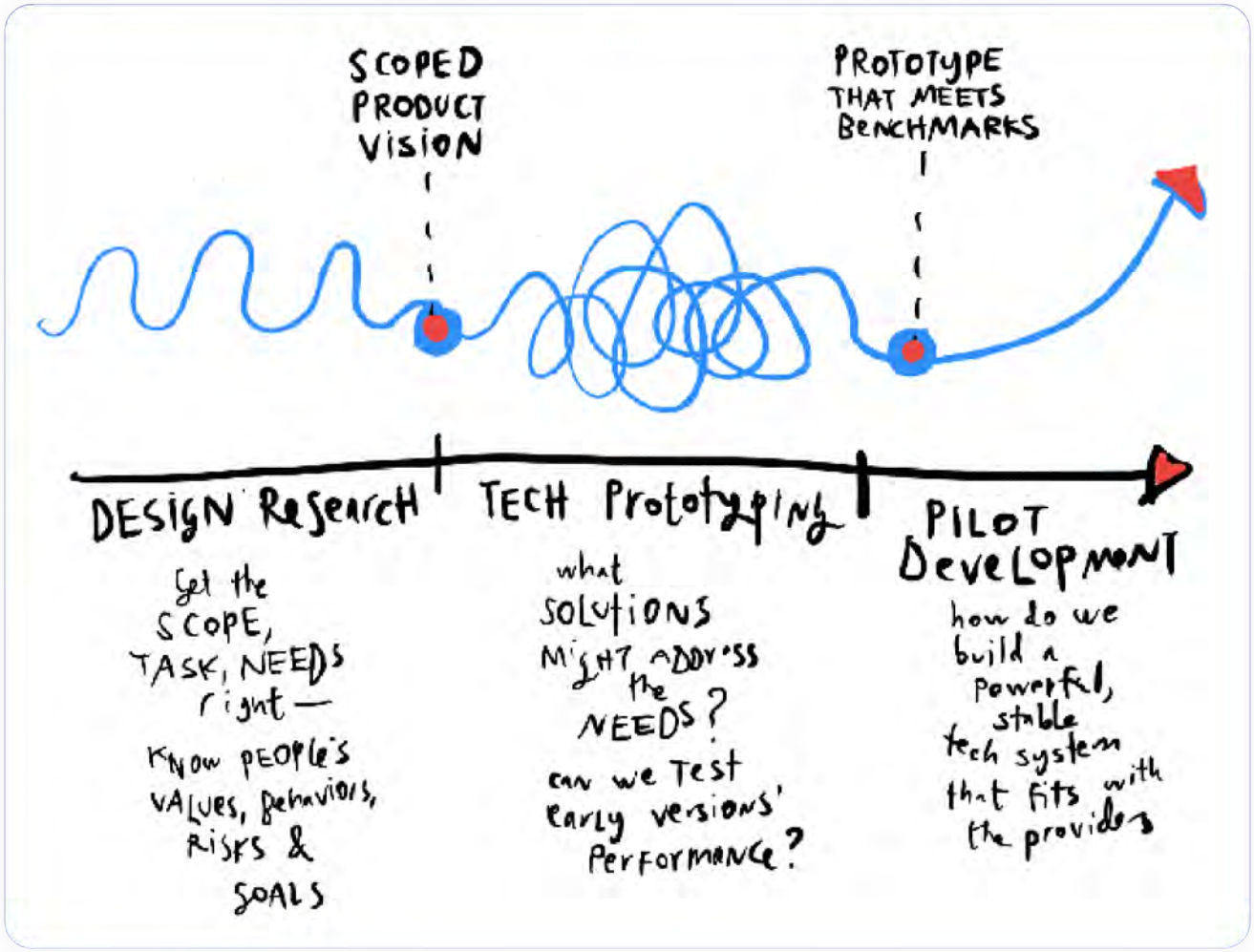
We knew that each specific legal aid project would need its own accuracy metrics, and its own specific safety plan, and its own way of being usable and intuitive for the use case it was for. So we gave each student team the same framework -- 'The 5 Gates for Pilot-Readiness' -- and worked with the students to customize this framework for their project area.

Gate	Project is pilot-ready if...	Example strategy to fulfill this gate
Performance and accuracy	Does the tool get the right answer, and is the output good enough to use?	Output checked against expert-reviewed ground truth, with known error rates rather than a demo that happened to work
Usability and equity	Can the actual target users complete the task easily with the system?	Real target users finish the task, across differences in language, literacy, and access
Human oversight and safety	Is a person in control of every decision that could harm someone, and does the tool fail safely?	Explicit review points designed in; the tool flags and hands off rather than deciding on its own
Adoption and change management	Can the organization actually deploy it, staff it, and fold it into the workflow?	A named owner, a training path, and a fit with how the work already runs
Sustainability and maintenance	Will it keep working, and stay accurate, after the builders leave?	Non-engineers can update the legal content, and running costs are accounted for

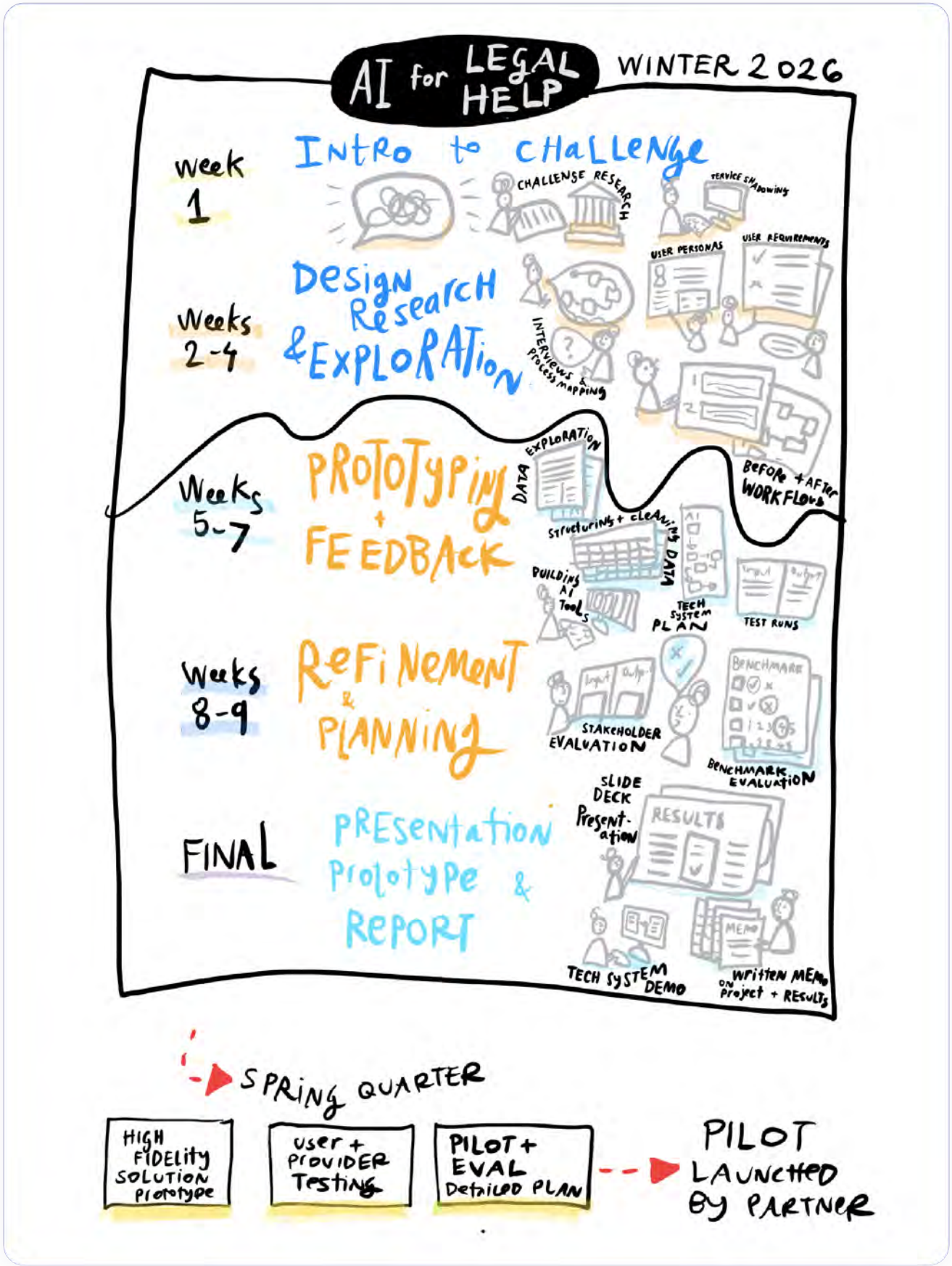
Where did these 5 gates come from? Our teaching team devised them based on watching the Winter Quarter teams talk through questions about ‘what is good enough, what will be safe enough?’ with their team partners. They also were informed by our conversations with justice leaders around the country about their own R&D in the AI era, and how they were wrestling with how to greenlight prospective projects. We found the 5 Gates useful to organize the multiple strands of evaluation needed to know if an organization is ready to move forward with an AI project. Reflecting, the first 3 (accurate performance, usability, safety) are likely the 3 gates most orgs should use to say ‘this tool is ready to try out with real cases’ -- and then the next 2 (adoption and sustainability) are more about ‘this tool is ready to be a full-time part of our work, moving out of early testing to regular use’.

With this 5 Gates Framework, each team built its own evaluation rubrics, protocols, and assets to figure out the pilot-readiness of their project. We wanted to have a semi-unified evaluation, so each team used this same framework but then could heavily customize how that gate is implemented in their situation.

Our class arc



The Winter quarter focused on forming the teams, and then scoping and building initial versions of solutions. Over the Winter quarter, each team learned its challenge in depth with help from their partners. They mapped the organization's current workflow and the vision of the future workflow the tool would create. Using Replit or their own technical knowledge, the students built a first-generation prototype to explore how a tool and generative AI performed on the given tasks.


AI for LEGAL HELP
WINTER 2026
week 1
INTRO to CHALLENGE
CHALLENGE RESEARCH
SERVICE SHADOWING
Weeks 2-4
DESIGN RESEARCH & EXPLORATION
USER PERSONAS
USER REQUIREMENTS
INTERVIEWS & PROCESS MAPPING
Weeks 5-7
PROTOTYPING + FEEDBACK
DATA EXPLORATION
BEFORE + AFTER WORKFLOWS
STRUCTURING + CLEANING DATA
BUILDING AI TOOLS
TECH SYSTEM PLAN
TEST RUNS
Weeks 8-9
REFINEMENT & PLANNING
STAKEHOLDER EVALUATION
BENCHMARK
BENCHMARK EVALUATION
FINAL
PRESENTATION PROTOTYPE & REPORT
SLIDE DECK Present-ation
RESULTS
MEMO
TECH SYSTEM DEMO
WRITTEN MEMO ON PROJECT + RESULTS
SPRING QUARTER
HIGH FIDELITY SOLUTION PROTOTYPE
USER + PROVIDER TESTING
PILOT + EVAL DETAILED PLAN
PILOT LAUNCHED BY PARTNER

Using design methods, we helped the teams move from a broad challenge to a specific vision of what a ‘golden path’ would be for this new system to a working tool. Teams interviewed their partners regularly, looked over synthetic or redacted data to see more about the current workflows, watched how the existing process actually ran, and identified where and how rules-based algorithms or generative AI could plausibly do useful work. Using all of this background work and collaboration, they started building in the middle of the quarter. By the end of the quarter all 5 of the teams had a functioning prototype at a live URL.

Winter quarter closed with a public presentation, to a room (and Zoom webinar) of attorneys, legal aid directors, court administrators, and justice technology practitioners to give feedback. Teams showed their scoping process, the before and after workflows, and their current running prototypes. The reaction from partners and audience members was specific and constructive. They were eager to learn more about the risks and failpoints, to understand what would be adaptable to other organizations, and to think through what resources would be necessary to operate and maintain these systems. One partner said that if the expungement tool worked, every legal aid organization in the country would want it. A national legal services funder pressed on cost and transcript accuracy. A state bar representative noted how useful it was to hear teams explain when and why they chose a rules-based system over a large language model, so that there can be more intentional choices about tech pipelines.

The materials from Winter quarter teams -- the live prototype, background memo about the R&D process, the standards and quality/safety rubrics, and the datasets -- were packaged to be handed over to Spring quarter teams. Some teams returned for the second quarter. In others, there were new teams or new team members to continue on with the work.



The Spring quarter was about testing and assessment -- though teams also had to use the testing results to refine (or pivot) their tech and design development. The premise given to the student teams was that the hard problem is no longer building something that works in a demo; getting that first version established is now a much easier task. Now we need to invest early and substantial resources into figuring out whether the prototype is accurate enough, safe enough, trustworthy enough, and integrated enough to put in front of real people facing high-stakes legal problems.

Spring quarter anchored around the 5 gates. Teams ran several rounds of user testing with the people who would actually use each tool, and structured review sessions with their partner, their team colleagues, and subject-matter experts. They built evaluation rubrics from the decision logic of their own tools, turned those rubrics into datasets and benchmarks, and generated synthetic test cases where they could not get real data. Each team built test suites that can be used to test the system for common users and inputs, but also the riskier edge cases, more complex cases, or 'harm floor' and 'red team' tests. The teams learned the limits of synthetic testing directly. Many teams initially worked to create lists of sample, synthetic users they could use to test -- but they often were too generic or non-representative to be a good stand-in for actual engagement with the target audience population.

By the end of the quarter, each team produced a pilot readiness assessment. This was an account of where the tool stood on each of the 5 gates, backed by the evidence of their tests, along with a draft pilot plan covering timeline, oversight structure, user recruitment, success metrics, and the governance and compliance work required before the system should be used in a pilot. The quarter closed with final presentations, again to an audience of practitioners and partners, with several partners joining remotely by video to see the progress of the projects and brainstorm how to make them stronger and scalable.

Findings and Takeaways

The 5 legal AI projects were different in subject, architecture, and maturity. We will share the teams' reports separately, in which readers can dive into the specific challenges and solutions. But we as the teaching team also identified common patterns and insights that we wanted to share with the field. We hope that others building AI and justice tools -- or funding and incubating 'build teams' can learn from these technical strategies, design insights, and collaboration techniques.

Where does AI fit, where not?

Start with minimal AI, add it only where it is needed, and make it show its evidence. One of our guest speakers mentioned this, and it came to be an important part of many teams' strategic work. When they scoped out the new technology agenda for the workflow, the strongest approach was to ask -- how much of the task could be done with plain logic, fixed forms, and deterministic rules? Where do we actually need generative AI, where nothing else that is more transparent and affordable would work? This is the opposite of the common instinct, which is to put a large language model at the center and build rules around it.

Two of the five teams deliberately chose not to use a large language model for their core logic. For the expungement screener, its core is a decision tree encoding Oklahoma statutes, and AI is confined to a single narrow classification step. Starting AI-minimal also keeps the system auditable, keeps sensitive client data out of third-party models wherever possible, and makes it clear exactly where the risk lives (since we can see the increased number of risks that come with AI).

We also learned that starting with rules can be a good first start, though it might turn out not to be sufficient. That initial rules-based prototype can reveal where relying on hard-coded logic fails and where generative AI might actually be necessary. The eviction notice team began with a pattern-matching detector, which forced them to formalize every legal defect in writing and kept client data local, but that detector hit an accuracy ceiling around 20%. Perhaps there was a role for a large language model? The team explored different tech setups, and found promise with a version with a language model bound by a structured per-defect rubric: for each defect, the rubric states the kind of check, the governing statute, the exact condition that should trigger a flag, the exact condition that should not, and the text the model must quote verbatim as its evidence. This hybrid approach could then resolve the concerns of auditability and maintainability.

The reconciled lesson is that “minimal AI” is the right place to start. But you don’t necessarily want to get stuck in all-rules, no-AI. Through the learning process, the team can learn to put AI only where the task genuinely needs it, and where you use it, constrain it so that every output carries the statute, a rationale, and a verbatim quote a human can check. Even if right now there is pressure to put an LLM at the center of everything, the skill teams built was knowing where generative AI earns its place as “the best solution for the task”. And the overall system should always force the AI to show its work and be checked.

Don’t trust the AI to remember the whole long conversation with the user: We found that constraining the model also means not trusting the raw conversation as the record of the facts. The demand letter team, whose chatbot interviews a tenant and drafts a reasonable accommodation letter, found that a long conversation could exceed the model's context window. That means that the caller might have requested something early, such as an emotional support animal accommodation. But then the model could (silently) forget this important fact, and not include it in the finished letter in favor of something discussed later.

The system produced a demand letter that still read as fluent and professional, which made it even more of a dangerous failure. Nothing looks wrong, but the document omits the very thing the person called about (it forgot about the support dog!). Then the letter might get sent out, without the key accommodation request that matters, and then the landlord is not obligated to do the key action the tenant wanted. The team's fix offers a general lesson: the system extracts the key facts of the case into structured storage as they are gathered, and generates the final document from that structured record rather than from the raw transcript (where pieces of information may be dropping away). The conversation is a helpful interface for the user experience -- but it is not the source of truth for key decisions and work product.

Fix and template the legal content, be flexible and generative elsewhere. The Motion to Set Aside team, whose tool drafts a court declaration, grappled with problems akin to those mentioned above -- generative AI that is not consistent enough, key information being dropped. Their work points to rules worth adopting by others in the legal help research & development space.

They made their system hybrid, not AI-first. Their tool's structured intake of the client's information and document assembly of the legal form do most of the work. The AI model is reserved for the one place it helps in a unique way: turning a tenant's messy account into a coherent, straightforward declaration about why they defaulted on their eviction lawsuit.

They also decided to lock the legal substance into the system, not letting the tool improvise or reconfigure this each time. The legal authority in the document, the citations and the legal argument, has to be fixed, attorney-reviewed, version-controlled templates that the tool selects among, never text the model generates. The system just pulls from these established, locked-in templates. In this high-stakes housing situation, free-form generation of legal authority is not safe. A hallucinated citation in a filing sworn under penalty of perjury could be catastrophic for the person's legal outcomes and housing stability. Anything the model does generate must trace sentence by sentence back to a fact the user supplied or a document they uploaded. Their lesson for the field: you should lock the law in templates, ground the narrative in citations back to specific shared info, and make every factual claim point to its source.

Getting the scope & workflow right

Scope to the task, not to the technology, and scope to the smallest end-to-end workflow you can ship. The teaching team had already worked with partner orgs before Winter quarter to scope a specific legal workflow: like eviction notice review, motions to set aside, expungement eligibility, and more. This scoping wasn't always sufficient -- at least for a 2-quarter, 6-month R&D cycle. Some teams needed to narrow down even further, to a more manageable set of discrete tasks to build and test for.

The teams that made the most progress through the 6 months narrowed their challenge to a single high-value task with a clear definition of done, and mapped the human workflow around that task before building out their tech pipeline. Other teams hit roadblocks with overly complex scopes, where an effective solution had to do multiple, interlocking combinations of tasks. Like with the Motion to Set Aside workflow, a full solution would have to do so much: interviewing the user, classifying them, creating legal documents, helping them file these documents, helping them serve other attorneys, helping them prep for the next step of a trial.

One of our guest speakers cautioned teams not to build a single-point solution that would only do a single task for the user, and then lead them to a cliff -- where they would then have to do the next step on their own (and likely fail). But it's also not possible to build out the whole set of interlocking functions and tasks that comprise a workflow in one R&D cycle. Teams needed to scope down to one part of the ultimate full workflow, to make it manageable. Then they'll need to do interlocking modules after they get that first one right.

Future builder teams might follow this heuristic for where to draw the line: scope to the smallest complete outcome the partner actually cares about. Get this part right, start seeing those important outcomes, and then spend more R&D time widening out to improve other parts of the workflow.

Invest time in understanding the big picture. One unintended consequence of the teaching team's prep of the scope before class was that some teams jumped straight into this specific part of legal work without seeing the bigger picture of how it fits into protecting people or raising claims. The student teams benefit from spending more time in understanding what the broader workflow actually is, what comes before and after it, and how it plays out in the field -- so they can rethink how they frame the problem and what tech they choose.

For example the eviction notice team jumped straight into the document analysis of eviction notices, because this was the scope given to them at the start of class. Only later did they understand what this document analysis really feeds into -- the production of the UD-105 Eviction Answer. Seeing it from that scope helped them rethink their tech pipeline and workflow design. It is an interview with a document analysis attached: the notice bounds what is eligible to raise, but most of the defenses come from other sources aside from the notice analysis (like talking to the tenant). A tool that treats the document as primary and the interview as secondary has the 'big picture' backwards.

In the future, we will work to balance this tight scope to students with more context and chances to do design research. The goal is to find places where the legal analysis is impactful, bounded enough to build, and clear enough about which part of the problem is technical and which part is legal or organizational. We want to make sure that the builder teams are building tools that match the shape of the real work and the legal and life outcomes that are most important.

Mapping the full current workflow is necessary: The voice intake team found that a key step of work was mapping the partner's existing manual workflow on paper and with the partner, in the first week or two, before any AI flow is designed. Several of the worst bugs, including a matter-routing misclassification and an uncaptured appeal callback path, came from designing handoffs into a process they had not fully charted or understood -- building before they truly knew all the ins-and-outs of how the current housing hotline operates.

Managing scope creep. It’s also very important to stop expanding the scope of what you’re building until the initial core scope is delivered. Even when the partner is asking for more. The demand letter team committed early to adding voice input (the person could get a draft letter over phone, not just on a website) because it was technically interesting and the partner was enthusiastic. But then they could not deliver it within the quarter because of an access dependency on a phone provider. Their lesson was that a single well-integrated, heavily tested feature that resolves the partner's actual bottleneck delivers more value than several loosely tested capabilities or new versions.

Ensure the workflow covers tech & data integrations: That team also surfaced a scope trap that is likely to come up in many future R&D cycles: how hard it might be to integrate with a partner's existing systems. Their partner wanted the tool to autofill information shared with the intake form, which sounded small but required coordinating with a separate case management system the team had not accounted for. Legal aid organizations care a great deal about how a tool fits into the systems they already run but they might not know what is and is not technically feasible. The burden falls on the team to map those system dependencies early and to flag where cross-system work would be needed.

How do we get high quality, safe solutions?

The human in the loop is a design decision, an essential part of many legal help workflows. Every team built explicit, purposeful checkpoints for human judgment before the system's output moved forward. This included attorney review of a generated letter, paralegal confirmation of a transcript, a volunteer's decision about whether to raise a defect, senior attorney escalation for an uncertain expungement determination. The teams designed these human-in-the-loop (HITL) moments into the workflow purposefully, with extensive planning with the partner legal teams.

There is no assumption that the tool will be safe enough in the initial pilot to go straight to the public or even to junior team members. The teams designed a human-tech workflow, including where those checkpoints sit, what they catch, and how the tool behaves when it hands off to a person.

The eviction notice team pointed this out: carefulness is a property that's not just about the LLM or tech pipeline choice. It is mainly about how the system (the model, the pipeline) is going to be implemented by human teams. A tool that is 70% accurate deployed with mandatory attorney review and sign-off might ultimately be safer than a tool that is 95% accurate deployed without that oversight. The right unit of analysis is the model plus the human team workflow, not the model alone. That reframe changes what a team should optimize, and also how they decide when a pilot is ready. It is why teams realized they had to design not just the tool itself, but also the oversight, like an audit trail workflow: every verdict, including the ones the tool cleared rather than flagged, that carries the governing statute, a written rationale, and a verbatim quote from the document, so that a supervising attorney can review a case in a few minutes. A safe system is one that is built to ensure the 'human in the loop' is supported as they review the outputs, even when they are at the end of a long day or under time pressure.

The voice intake team turned this HITL principle into a key design choice for their hotline prototype: they removed the tool's ability to reject anyone outright or immediately. An initial prototype version could tell an eligibility-screening caller, at the end of the call, that they did not qualify. They replaced that outright rejection with a tentative eligibility flag that a human reviews, and eliminated end-of-call hard rejections entirely. The change did 2 things. It stopped a distressed caller from being turned away by the system, before any person had looked at the case. It removed the incentive for callers to feed the system false information to get past this rejection gate. The general pattern is that an AI serving vulnerable people should not be allowed to make an unrecoverable negative determination on its own. It flags, and then an expert human decides.

Legal tools' stubbornness versus usability. One partner surprised a team by praising the chatbot's stubbornness in pushing a user for complete answers, a behavior the team had treated initially as a flaw. It might not be the most user-friendly when a chatbot refuses to let a user move to the next step before it fully engages on a point -- but that might be the safer, more responsible choice. In high-stakes documents, an AI that refuses to settle for an incomplete answer is doing the right thing.

User experience and legal limits might come in conflict. Sometimes good user experience can have negative legal consequences. The demand letter team found a choice that looks like a user experience preference affected the legal output the tool produced. A prototype of their tool let the tenant choose how long to give the landlord to respond, which seemed like ordinary configurability. But then in testing distressed users began demanding unreasonable windows such as changes being made in 24 hours. The partner pointed out that the response period is a legal matter, not a preference. The testing and partner input helped the builder team remove this from being a user choice and standardized it.

A few lessons come from this pivot. One, even when trying to build a user-centered tool, do not hand a distressed person a choice that they might use against their own interest or that might jeopardize their legal claim. Also, double-check and route design decisions past the partner because some of the 'user experience design decisions' are legal decisions in disguise.

The team also learned that having the humans review just the end-product letter might not be enough. Harms or problems might also happen in the conversation chat leading up to the letter. There needs to be review and oversight of these conversations, because things can happen in this back-and-forth conversation that might hurt or mislead the user. What if the tool drifts into giving legal advice, or gathers too much high-risk data, or does something else (that might not show up in the ultimate letter being reviewed). There needs to be review and backstop plans for all the interaction parts.

Too much warmth and empathy for the user can knock the tool off course. The demand letter team's user population is largely in distress, so the team leaned into empathetic language to make the chatbot feel humane. Then feedback from the field pushed back: too much warmth can lead a user to treat the chatbot as a person, or to lean on it for emotional support it is not equipped to give, which can pose its own kind of harm. The resolution they reached was about finding a balance between empathetic support and clinical legal assistance.

The tool can acknowledge feeling empathy briefly and sincerely, along the lines of noting that a situation sounds difficult, while staying focused on the task and declining to offer the user any medical, legal, or personal counsel. Empathy in these tools is a design dimension in its own right, distinct from accuracy. It can help with usability, but it also could be mis-calibrated too far in both directions. Too cold and a frightened person disengages, too warm and the tool invites a reliance it cannot live up to. How much warmth, and where it stops, deserves the same deliberate calibration as any other part of the system. It will be different depending on the workflow and who is hosting/running this tool. It is worth testing rather than guessing.

Sometimes the user's ability to review for accuracy is just too limited. We challenged the students to be honest: is AI or any other tech tool really a good fit for the workflow at all, or would it be better to have a fully-human workflow? The Motion to Set Aside team had to grapple the hardest with this question. Their intended user was a self-represented tenant days from a lockout. There are not enough lawyers in the region to serve people in this situation, especially since it's unpredictable when a person might want to raise this claim and then they need intensive, emergency help to do it. Because of the lack of human staff, the challenge was whether there could be a new tech-powered workflow that would have the tenant themselves interacting with the tool and acting on its output.

The team came to see that it's unlikely that the self-represented tenant can do this on their own, especially when it comes to deciding if the tool's output is good to go. A stressed-out non-lawyer does not know what to verify, cannot recognize a hallucinated citation, and cannot tell whether a sworn declaration meets the burden of proof. The team's red-teaming showed that stressed users tended to over-trust polished output from the tool. This wasn't an edge case, it seems to be a predictable trend of over-reliance on conversational AI's output. The conclusion was to design a workflow on the assumption that the user cannot check the work on their own. This means that a person can't use the system by themselves, at least in the current workflow design. Ideally, a trained intermediary has to be the human in the loop, before the tenant uses anything the tool produces.

The team pivoted -- moving from a public tenant-facing app to a supervised, staff-facing tool. The team drew up a principle about when a tech tool is ready to release: better than nothing is the right bar for measuring impact, but the wrong bar for deciding to release. The fact that the alternative is no help at all can justify building the tool. But it cannot justify shipping an unsupervised version whose failures (like user over-reliance, and inability to check for accuracy of high-stakes documents) are predictable.

Human review is important for professional development and quick fixes of mistakes. The expungement team, whose tool screens eligibility, also framed the human step in the workflow very sharply. They realized that the tool is helping the attorney keep their expert human judgment in the loop for every ambiguous determination, making errors recoverable and auditable. An attorney who overrides a wrong call and records why they did so produces an auditable trail of their decision. This is a different world from a tool that emits confident wrong answers with no correction path.

It’s not enough just for a system to be correct. Its output has to be legible enough for the reviewer to verify quickly. With this tool, the partner did not want to use an early version that gave verdicts without showing the statutory reasoning behind them. They wanted the team to be citing back to the grounding law as a way to reinforce this expertise with the human user, and also to assist in the review. The team also found that sometimes the system was giving the reviewer warnings that were too 'soft', especially when a possible misinterpretation or incorrect output can affect a person’s high-stakes outcome. The team added in disclaimers with ‘hard stops’ so a person has to read it and acknowledge it. The goal was that juniors or in-training team members would be more likely to perform the safer behavior.

Speed and accuracy often are in conflict, and different projects will have to favor one over the other. Two teams reached opposite-sounding conclusions about building a system to be either faster or more accurate. The eviction notice team, building a document-review co-pilot with a supervising attorney checking every output, is moving to a system that prioritizes efficiency over complete accuracy. The goal is that for a frontline partner in the courthouse hallway, under deadline pressure, a tool can do a quick preliminary pass and then encourage the human team member to do thorough human review of the output while going through a client interview. The goal is to get a quick pass over to the human in the loop, for them to review and judge how to use.

The voice intake team, building a phone system that screens callers for eligibility, moved toward a system that prioritizes accuracy over speed. Their ultimate prototype focused on never turning away an eligible caller rather than keeping calls short. They made this choice because a wrong rejection ends the call and excludes a real client before any person has looked at the case. They wanted to make sure a person was never rejected inaccurately, in the name of keeping the system 'quick and efficient'. They also discovered in user testing that a caller who senses the system is about to reject them will start feeding it false information to qualify. This led them to prioritize accuracy even if it made the call a little slower or longer.

Both teams made a call on speed versus accuracy working with their partner and the tool’s context. The key thing is to find the failure that cannot be undone. Where a competent human reviews the output quickly and can catch a mistake before it reaches the client, then a tool might optimize for speedy throughput. Where an error is categorical and unrecoverable in the moment, such as a rejection that ends the phone call interaction, then a tool might optimize for carefulness and route the decision to a human. Speed might get sacrificed. The design question is not speed or accuracy in the abstract. The main job of the builder team is to find where a mistake becomes irreversible, and whether an expert human reviewer is there to catch it or not.

The expungement team refined this a step further. They identified the parts of eligibility screening where speed matters most, like the rote statutory lookups, the charge classification, the waiting-period arithmetic. These are exactly the parts where deterministic rules are most reliable. Then there are the parts where care matters most, like multi-county sequencing, ambiguous charge language, borderline categories. For these, the team added friction on purpose through hard-stop warnings and required acknowledgment. In some parts, the tool prioritized speed, other times there were intentional slow-downs.

The workflow was not all fast or all slow, it adjusted depending on the risk level and type of tech being used. The design move is to map the workflow, decide segment by segment whether speed or care governs, and build fast paths and deliberate friction accordingly.

What partnerships, outreach, and collaboration are necessary to get a successful project?

In this field, you have to have solutions that are multi-stakeholder. If our projects were commercial products, then we might say that it is the end-user who should decide if a tool is ready for pilot and will likely be adopted. With legal help AI tools, it's more complicated. It's not just about whether the end-user finds it valuable and engaging.

- A supervising or managing attorney is likely the one that decides whether the tool is adopted in a team,
- A frontline attorney, volunteer or paralegal will be the one using it
- A client bears the consequences of a mistake, or gets the benefit of a successful output

Those are 3 different stakeholders, with different sets of values and standards. The features that make a volunteer's day easier, such as fewer clicks, are not the features that earn an attorney's trust, such as an explicit audit trail. If teams designed for just one stakeholder, the solution is likely to fail.

Several practical consequences follow from this multi-stakeholder challenge. The partner relationship is not a single relationship. Teams need to meet with people from different backgrounds and roles. Teams that met regularly with both the managers who set priorities (leadership POV) and the frontline staff who know how the work runs (frontline POV) caught wrong assumptions early, got accurate test suites, found the right workflow maps and scopes. Hearing only from a manager produced tools that looked right on paper but then would ultimately hit a wall -- when the team realized they were building something that the frontline staff wouldn't have time or need to use.

Then again, hearing only from frontline staff produced tools that solved a real daily pain but did not fit the organization's constraints or ultimate goals. The team might build something with little payoff or that wouldn't be given further resources.

Know who makes greenlight decisions. It's also very important to figure out which person or group of people have 'deployment authority' early in your project. One team spent two quarters assuming their main partner contact could authorize a pilot, and learned near the end that the decision to deploy sat elsewhere in the organization, with someone who had not yet been brought into the development work. These design choices (and buy-in moments) were made under a false assumption about who could give a greenlight. The named project contact (who might be excited about AI and innovation) might not be the person who can put a tool in front of a real client or get the organization to commit to a pilot. Builder teams should ask in the first meeting about how to get the right 3 stakeholder groups involved, and what standards and expectations they each have.

Be ready to talk through hosting and maintenance. Some teams began their development cycle with the assumption (and hope) that their partner organization would host and own the tool. As they moved along with conversations, they found that hosting and real ownership of the tool would need to be arranged separately. They had to substantially adjust their rollout plan. The partner frontline legal aid attorney is an invaluable source of failure modes and domain expertise, so likely the lead stakeholder contact early in the R&D work. But you also need to find the people to involve, who have the authority to commit an organization to hosting, IT, and maintenance. The team can treat the frontline expert as the domain expert and the red-teamer. But the institutional owner (the person or unit that would commit to hosting, legal review, onboarding, and maintenance) is a separate contact to establish early.

Frontline experts should be major contributors. It cannot be overstated how important it is to have professionals who have practiced the given workflow day-in, day-out with many different types of client. With voice intake, the partner was a substantive collaborator, and their operational judgment shaped the architecture directly. The partner had important insights about what would work well in practice and what would cause problems. They rejected emailing intake summaries to callers on privacy grounds, they rejected live web scraping of eligibility thresholds in favor of a folder staff update by hand, and they rejected a web-chat design out of knowledge that the organization's callers would not use it. Each of those decisions was borne out by later testing.

A builder team working with a public interest partner should assume the partner has accurate views on how the system should be built, how people will behave, what safeguards need to be built in, and what functions are likely to go unused or misused. The team needs to structure the project to take in this frontline knowledge early. The voice intake team also learned to confirm operational understanding early rather than late. The bugs they found later traced to gaps in their knowledge of the team’s work and rules. Ideally those could have been caught in week 2 with conversations, rather than extensive bug tests in week 6.

Partners' time is limited and might stall R&D. Building in this collaborative way is necessary, but it also can be a slowing factor. Several teams found that the partner's review bandwidth, not the team's building speed, set the pace of the progress they could make. When an expert can review only one batch of test cases a week and a supervising attorney can review one a month, the teams might feel stuck -- but they cannot responsibly move forward without this expert input.

Public interest technology is a coordination and trust problem with a technology layer, not the reverse. The tools that had a path forward were the ones where the partner organization was very involved, correcting the legal logic, judging the output, and telling the team what would survive real-world usage with their staff and clients. The technical build was the smaller part. The larger part was the relationship: repeated review sessions, regular reporting of what did not work, and design decisions made with the people who carry professional and ethical responsibility for the outcome. This is slow, in-person work, and it does not end when the first pilot version launches.

The expungement team called out a structural feature of this kind of development work: the people ultimately most affected by a tool's failures are usually not the people who can report them -- there is too much time and organizational distance. A client never touches an eligibility screener or gives feedback on it. But they feel its error months later, downstream, when a petition is filed and rejected and the window to fix it has closed, and by then the failure is nearly impossible to trace back to a specific output. The feedback loop that would catch the problem in a commercial product is broken here, because the affected party is not the system's direct user.

For builder and partner teams, before embarking on tech development, describe with real specificity what a bad outcome looks like for the client, not just for the attorney and not just for the organization. That user-centered description of harms should be prominent enough that it changes design decisions. Builder teams can ask these 2 questions early in the cycle to raise and center these possible user harms:

1. What case or prior tool failure would make the partner unwilling to proceed at all? What are those down-the-road harms that would leave a person worse-off? This will surface the threat model faster than just having a general conversation about risk.
2. What would have to be true for the partner to use the tool without the builder team in the room? How would they feel comfortable that they can adequately use and oversee it? This will define the minimum that the documentation, the onboarding, and the human-review design have to reach.

Authority in the R&D process should follow accountability: the partner who holds the professional liability and the client relationship is the one that decides what safe and good mean.

How do we build with safety as a core goal?

Safety is a main goal, but it also can take many shapes. In many commercial AI products, a tool is good if it creates value for the business, and safety is a constraint on that value. But with legal help tools, teams found that being safe is itself a core objective. It's not just a cost to trade against efficiency. Success for the partners means the tool is both good and safe for everyone it touches, which includes the organization's own staff as well as its clients.

That said, each team needed to find exactly what 'safe' meant for their workflow and organization. The teams found that safety splits into distinct kinds that need different treatment.

> **Output safety** is whether the tool produces content that could harm someone if they act on it, such as a wrong eligibility determination or a hallucinated legal fact. Measuring it belongs with how we evaluate performance and accuracy.
>
> **Interaction safety** is whether the tool handles a vulnerable person appropriately in the moment. Does it recognize a caller in crisis, and then escalate to a special path rather than continuing a hundred-question script? Does it get human oversight at key moments?
>
> **Consent and disclosure,** meaning whether the user knows they are talking to AI and understands what it can and cannot do, is a safety concern crossing both outputs and interactions.

Safety work, planning, and evaluation needs to treat these 3 different dimensions separately.

Some concrete safety practices came out of the one team's work and can transfer to any other builder team. Data minimization should be the default approach. The team collected demographic fields at intake because intake forms collect demographics, then removed them after a mid-quarter self-audit found the tool was storing tenant name, address, contact information, and income in its database. The better default is to justify every field before it exists, not to audit fields after the fact.

It also pointed to a key practice of builder teams -- to run audits and make sure what you are claiming is based in the deployed code, not just in your design or planning documents. The team might have aimed to build a tool that saved no data, but what if the running code does not match that description? Safety reporting has to describe the system that is actually running, not the system the design document intended.

The team also was careful in listing and profiling its tool's failure modes. The team catalogued every risk into 1 of 3 categories:

- one the tool mitigates,
- one it inherits from the manual process without making worse, or
- one it introduces that did not exist before.

That three-way distinction separates the risks a tool reduces from the risks it carries forward and the risks it creates. This same kind of failure mode cataloguing and categorization could transfer cleanly to an intake assistant, a know-your-rights chatbot, or an eligibility screener.

Aside from safety around the client, teams realized they also had to have a safety plan to protect the frontline worker. The voice intake team's system correctly de-escalated hostile callers and transferred them to a person, but then the team realized that handing an angry caller to a staff member with no warning exposes that staff member to abuse. Their fix was to flag hostile or highly distressed language in the transcript handoff so the human on the other side knows what they are walking into and can plan the callback. Designing for staff safety, not only client safety, is part of the work in making a system that truly is safe enough to pilot.

Guardrails themselves can fail. A visible failure often might be hard to understand and fix. The Spring quarter demand letter team inherited a prototype with a safeguard that ended a conversation when a generated message hit certain length and content conditions. It worked most of the time but then occasionally fired early, cutting off the interview before all the information was collected, which then produced an inaccurate letter. This kind of crude guardrail can introduce its own failure mode, which isn’t always obvious. The team observed a bad letter output, and had to hunt upstream to find what was causing it. Sometimes the cause looks unrelated, like this length termination rule. The team had to learn to trace a failure to its real cause before fixing the symptom.

Evaluation is hugely important now that building is so quick

Prototyping tech tools has become incredibly fast. High-fidelity iteration is the new cycle, with lots of evaluation baked in. The vibe-coding platform Replit was excellent for getting teams started with building tool prototypes. It let students with no engineering background chat with the coding agent and stand up a working, interactive, AI-powered tool in days. They could take the scope and goals from their design research, and create first demos that were genuinely impressive. But getting from that first build to something worth putting in front of real users took repeated cycles of hands-on design work. That is where teams moved to higher-fidelity design and prototyping tools and spent most of their weeks during the 6-month cycle.

The teams still had a long tail of expert review, red-teaming, bug discovery, and user testing. This hard work suddenly became where the real time goes -- not the technical development. There is a big, hard-to-close gap between a prototype that works in a demo and a tool an organization will stand behind in production. It takes months if not years to close this gap. Anyone budgeting a legal AI project should budget for the tail, not the prototype.

The voice intake team, in their class debrief, flagged a specific warning about what the fast first build actually is. Their prototype was built on Replit with heavy use of AI-generated code. They came to treat it as a working specification of what the system should do rather than as code anyone could deploy. It helped in defining a clear functional agenda of the new workflow, but it was not stable or safe enough for pilot. Before real clients touch a system like this, the code needs a professional engineering review for security, error handling, and maintainability. The team flagged that this must be a named gate in their rollout rather than an afterthought. The prototype proves the behavior. It is not yet the product and shouldn't be put in the field.

The experts do not have premade evaluation checklists. A recurring surprise was that the attorneys and subject-matter experts who partnered with the teams did not have ready-made standards for legal accuracy, sound practice, or safety that a tool could be measured against. The student teams hoped for the frontline experts or organizational leaders to tell them exactly what standards to meet. But there was no checklist to hand over. The knowledge lived in the experts' judgment, built over years, and it had to be drawn out case by case and written down before it could become a test.

The builder teams spent hours of time sitting with partners, walking through examples, and turning "I would know it if I saw it" into specific, observable criteria. This creation of evaluation standards and rubrics is slow but necessary. It is one of the most valuable things the course produced, because the written-down version of quality rubrics, safety standards, harm floors, and test suites are what another organization can pick up and reuse as they build solutions.

How best to build these evaluation rubrics? The voice intake team built their evaluation rubric themselves and validated it with the partner's paralegals afterward, and concluded that this order was backwards. A rubric encodes whose judgment counts. If the people who will supervise the tool in production are the partner's staff, their judgment should shape the rubric's structure from the start rather than confirm it at the end. Co-developing the criteria with the supervising staff, beginning early, is how a rubric ends up measuring what the organization actually cares about.

The 5 Pilot-Ready gates emerged from the work. The pilot-readiness question, and its five parts, were not imposed from a template at the start. They surfaced over Winter quarter, from watching different projects run into the same kinds of trouble, and from asking the partners what would have to be true before they would trust a tool with a real client. Performance, usability, oversight and safety, adoption, and sustainability are the 5 categories that kept reappearing as the big ones. Pilot readiness is not a single yes-or-no question.

Evaluation matures from watching to measuring, with good test data as a common bottleneck. As we worked on different kinds of evaluation over Spring, the teams started with qualitative observation, meaning watching a few people use the tool and noting what broke. They gradually moved to systematic testing against a rubric built from the tool's own decision logic. Both stages are necessary, and likely this order was a good one to follow. You cannot write a good rubric until you have watched real use.

Access to 'Test Suite' data is a huge, burdensome need. The build teams need real data (or very-close-to-real synthetic data) to evaluate the quality and safety of their tool. The hard part of the measuring stage was not writing the rubric. It was getting test data. Teams needed sets of annotated documents or case examples that represented the real spectrum of situations and inputs a tool would face: the clean case, the messy case, the rare edge case, the case designed to break it. Real client documents were mostly off limits for privacy reasons, so teams built synthetic ones, and building a synthetic set that genuinely covers the range of scenarios, with correct annotations for the right answer in each case, turned out to be difficult and slow. A rubric is only as good as the cases you test it against. But getting access to representative data was a huge block.

Get settled expert standards first: The teams' evaluation work also flagged an important step any benchmarking effort should begin with: measure how much your human labelers agree with each other before you read any disagreement between the tool and a human as tool error. One team had 2 experienced reviewers label the same synthetic notices, and the 2 reviewers disagreed with each other about as much as either disagreed with the model, with agreement statistics low enough that no single reviewer could serve as a clean answer key. In other words, part of what looks like model error in legal document review is really the absence of a settled ground truth among practitioners. Even the human reviewers don't agree with each other. Further accuracy gains at that point require expert adjudication of the disagreement set, not better prompting.

The practical sequencing lesson the team drew is to build the evaluation harness before the tool: encode the legal tests as checkable specifications, build the benchmark, measure human-to-human agreement on it, and only then build the tool that has to beat it. The team's first benchmark and their agreement measurements arrived after the prototype they were meant to judge, which meant their early confidence was intuition rather than evidence.

Sequence of testing: The voice intake team added a staged testing arc worth copying: synthetic personas first, then live volunteers playing those personas, then the partner's own staff testing in their real operational setting, then adversarial red-teaming. The progression runs from controlled to realistic to hostile. Each round caught failures the previous rounds had missed.

The decisive round was partner-staff testing. When the partner's paralegals tested the system, they surfaced a collapsed Spanish-language flow, a matter-routing error, and a premature eligibility rejection that no synthetic run and no peer testing had caught. Those staff tested the system with cases that it would actually face, based off of their frontline expertise. They interacted with the system in the way that the work actually happens. The team concluded that they should have put a real tester in front of the system by the third week, well before it felt ready, because the bugs and problems a real person surfaces early are far cheaper to fix than the same issues surfaced late.

Pilot-readiness gates likely need to be staged, one-at-a-time. We also realized that the five gates are not parallel tracks but a dependency chain. Usability cannot be judged before performance is confirmed, and safety cannot be judged before the underlying logic is verified. One team tested the gates in parallel, hit a rule-engine error during usability testing that forced a second round of accuracy testing they thought was finished, and lost about two weeks. Sequence the gates, and grade the later ones only once the earlier ones are performing well.

How to build a good pilot-readiness rubric: The voice intake team produced several transferable lessons about how to structure a readiness rubric, each one growing out of the team watching their initial rubric fail. First, categorical failures belong outside the scored rubric as binary pass-or-fail gates. This needs to be treated differently than other rubric items. Turning away an eligible caller or skipping a required disclosure should not be averaged in with voice quality, because a smooth call that misses a required step is not a good call, it is a failure that happens to be smooth. Second, equal weighting is a hidden assumption that is usually wrong. Scoring every dimension the same silently claims that voice quality matters as much as eligibility accuracy, so weights should be explicit and tied to the real stakes of each dimension. Third, a rubric should output an actionable recommendation, not a bare pass-or-fail signal. The question a partner actually needs answered is not whether the tool passed but what they are allowed to do with it now, so the score should map to a deployment scope and a supervision level rather than to a single verdict.

User testing results versus real-world behavior. Some of the user testing run by teams was done by participants playing out the scenario of a fictional user, but they weren't actually in a stressful legal problem. The teams warned that comprehension in a lab-based user test does not predict behavior. Their educated testers, role-playing tenants while sitting in a calm university classroom, understood the tool and still appeared ready to trust its output without checking it. A user who can explain what a tool does, or who finishes a task in a low-stress session, tells you the interface is legible. It does not tell you about what's going to happen in a real pilot. This user testing won't predict what a frightened person days from a lockout will actually do, which might be to accept polished output without question. Measuring understanding is not the same as observing behavior under real conditions. Teams have to factor this safety risk in, and find other ways to test for it.

Find non-expert testers. The expungement team added two more evaluation lessons. The first is to test with non-specialists, not only the experts who helped build the tool. When specialist attorneys ran real cases through their screener they moved through it without trouble, often because they were so familiar with the legal workflow and this tool itself. But attorneys who do not practice in the area got stuck immediately, because the interface assumed a familiarity with reading a criminal-history report that the occasional users did not have.

The real user population for most legal aid tools is not the dedicated specialist legal help team that co-designed it, who have extensive subject matter expertise. It is the broader set of attorneys or justice workers who work on these cases occasionally and need the tool to provide its own instructional context. Testing only with the specialist produces something that works only for them.

AI Projects Need Intentional Sustainability Plans

Design for maintenance by the people who know the law, not the people who know code. Sustainability was the gate that teams scored lowest on, and felt the least prepared to meet. Often sustainability and onboarding only feel urgent when deployment is near, which is exactly when there is no time left to address them. The teams that thought about it early converged on a clear pattern about maintenance. The law changes, ordinances turn over, and clinic practice shifts, so the only ongoing maintenance task a tool should ask of legal aid staff is editing plain-language descriptions of the legal rules. They should not be expected to edit code or LLM prompts. The eviction notice team built toward a design where each legal defect is a short paragraph of English stating the rule, the statute, and the conditions that do and do not trigger it, so that an attorney or paralegal can amend the tool's behavior by rewriting a paragraph.

The benchmark is what makes that rewriting safe. When a legal team member rewrites a tool's rule, the system is re-scored against the labeled test set before it can ship. A legal expert can evolve the system and a regression is caught automatically. This also argues for keeping the legal content in versioned data files rather than buried in code, and for keeping the model layer provider-agnostic so that swapping in a better or cheaper model later is a configuration change the benchmark can immediately evaluate. A tool that requires a developer and a redeploy every time a statute changes will not survive the team that built it.

The voice intake team reached the same content-not-code conclusion from a different angle. They shipped a folder where authorized staff upload the annual poverty guidelines that drive eligibility. In that way, a threshold change is a document upload by a legal team member rather than a code change by a developer or a live web scrape that could hallucinate a wrong number. They also named a discipline the maintenance conversation usually omits: monitoring is a design input, not a downstream deliverable. The question of how anyone will know the tool is working in production should be asked in the first week, because the things worth watching, such as call duration, escalation reasons, and caller feedback, are far easier to instrument into a system while it is being built than to retrofit once its behavior is fixed.

The expungement team had concrete ideas to address this challenge of changing legal rules. Because the statute they encoded was actively changing, they designed an admin dashboard that exposes the eligibility rules to legal staff as editable plain-language parameters, such as waiting periods, thresholds, and disqualifying-offense lists, each with a last-updated date and a versioned change log recording who changed what and under what authority. Saving an edit automatically re-runs the synthetic regression set, so a specialist can confirm the change introduced no errors before it goes live. This puts day-to-day legal maintenance in the hands of the people who own the law and reserves engineering for genuinely structural changes. Their broader conclusion: treat the partner's capacity to maintain the tool after the team leaves as a design constraint from day one, not a later deliverable. A tool that works at handoff but cannot be sustained by its users is not a finished contribution.

Sustainability also has a financial dimension that student teams tend to underweigh, because they build on free tiers and do not see the bill for the tool. The demand letter team named this as a gap in their own work: they had not analyzed what the tool would cost the partner to run, from model usage to hosting. For an under-resourced organization, the ongoing cost of operation is a real deployment question, and a tool that is too expensive to run is as unsustainable as one that is too hard to maintain.

Next Steps: Coordination, Scale, Commons

As we watched the five teams and their partners work on solving very specific legal help workflow challenges with technology, we as the teaching team were thinking about how this fits into the larger access to justice ecosystem. While we watched each of their R&D journeys, we were thinking: how can we make all of this work product reusable, useful, impactful? How can we help other organizations who want to similarly transform their eviction defense, reentry, and other legal work, who are not in this room?

The Legal Help Commons grew directly out of this class. The students' thorough development work and documentation produced valuable material that the field doesn't usually produce or share: the rubric for what a tool should do, the honest list of how it failed, the benchmark to test it, and the architecture decisions behind it. Watching five teams answer the same underlying questions five times made the larger problem plain. Across the country, organizations rebuild this same scaffolding before they can build the part that is actually local to their jurisdiction. The Commons is the response: build the common parts once, in the open, and share them, so every team starts from a tested standard instead of a blank page.

Each project will be featured on Legal Help Commons as a package of development materials, in addition to a single report. A package gathers everything another builder or frontline team needs to build a given kind of tool and everything a leader or funder needs to evaluate one:

- a functional agenda that defines what the tool must do,
- a conformance standard that sets the bar it must clear before a pilot,
- an evaluation protocol and shared test suite,
- a reference architecture,
- an open-source reference implementation you can request under license, and
- the case studies and results from the original build.

Each package lives on a workflow hub page that assembles these assets in one place and marks each one honestly as ready, drafted, or not yet started.

The point of featuring the work this way is coordination and instigation. The packages are open and unfinished on purpose, published as drafts from real builds so the field can test them, argue with them, and improve them. Anyone can read the agendas and standards to scope their own build or evaluate a vendor, join a cohort where the standards actually get decided, or contribute their own specifications and test results so the next team starts from them. The class produced a set of useful tools. The Commons is how those tools become a shared foundation the next teams build on, building together rather than separately.

FIND THE PACKAGES

legalhelpcommons.org

Legal Help Commons

CONTACT

Stanford Law School
Legal Design Lab

legalhelpcommons.org

legaldesignlab@law.stanford.edu