Craig Zilles: I've been using PrairieLearn for a long time, so I'm user Number 3e for PrairieLearn. Now there are hundreds of thousands of users. I want to say I've been using it for about 15 years, and then the computer based testing facility, we started at Illinois about 12 years ago, and so that started with my class. I hired the first proctors I have a bunch of experience, and so I'm going to try to share some of that with you, but it's hard to remember what it is that I have the expert blind spot, so to do ask questions as they come. I'm going to focus a bunch on PrairieLearn. PrairieLearn is open source software, and it was built primarily for this top purpose, which is to enable mastery based homework. We're strong believers in this notion of mastery oriented teaching of, like, you should keep trying something until you get it right that having a bunch of homework problems which you try and then submit, and a week later, you get feedback about whether you got it right or not, that's not going to maximize learning. What maximizes learning is being able to give you immediate feedback on, did you do the thing correctly or not? Then immediately giving you another opportunity to try the thing. PrairieLearn has really good support for auto grading, and so that enables us giving students immediate feedback, and then the ability to write question generators allows students to say, give me another version of the question until they're satisfied that they can do the thing. A few years after the PrairieLearn existed, we started running asynchronous exams in our computer based testing facility. Asynchronous means that students are taking the exams at different times. This turns out to be super important for the reducing the administrative overhead of exams. There's a non trivial amount of work of students having conflicts and getting sick and dealing with all that, that if you're just able to say, hey, sign up for a time that works for you, a lot of that overhead goes away. But importantly, we can't be giving the same exams to students taking the exam at different times because one student will tell the other students what's on the exam, and so having question generators and pools of questions enables us to generate randomized exams. Also, auto grading facilitates the use of frequent testing that basically, what the CBTF does is it allows you to outsource the proctoring of the exam, outsource the scheduling and administrative overhead of the exam to Prairie test, and then if you can outsource a bunch of the grading, it makes running exams very cheap. If exams are cheap, then you can deploy them in ways that are pedagogically useful. We find that if you run many small exams, the students end up learning more and are less stressed. If you run retake exams, if you give them an opportunity to take an exam, fail, figure out what went wrong and take another version, they're willing to do the work to learn the material. I think this is especially important now in the context of GenAI because the students are confusing themselves about how much they're learning when they're working with AI tools, that many of us at the beginning of the semester have gotten up and said, here's how you should study for my class, and you tell them, here's all the best practices. Most of the students go, yeah, whatever, I'm going to go do my own thing, and that's fine. They're optimizing their lives. But many of them misunderstand what they need to do in order to succeed in your class. Having the opportunity to do whatever they think is going to work, have that fail, and then have an opportunity to listen to you about what actually will work and get a bunch of those points back by demonstrating that they actually did learn those things on a second exam is really important. Being able to run these exams asynchronously with auto grading, at least a significant amount of auto grading, enables faculty, even in large classes to run many versions, many exams and retakes. As much as possible, we want to cover all our learning objectives, that PrairieLearn was originally built as a reaction to the questions that learning management systems at the time and even Canvas now can ask, as well as what book publisher homework tools we're providing that we weren't satisfied that it could do all the things that we wanted tested, and so we didn't want to dumb our classes down by being constrained to what a given tool could do, and so PrairieLearn has a lot of flexibility and extensibility that allows it to not dumb down your course. What PrairieLearn is trying to do is be in the sweet spot between what LLMs like Canvas can do, which Canvas makes it very easy to write a small set of questions. On the other side of the spectrum is you could build a tool from scratch that does whatever you want. That's very flexible, but it's also a lot of work. What PrairieLearn is trying to do is be somewhere in between on this Praido optimal surface where we get most of the flexibility of being unconstrained platform, but also we can do a lot of things very easily. I'm going to show you some examples of that, but

there's a couple of key concepts in PrairieLearn. One is that maintaining a collection of questions is very much like a software engineering exercise, and so we should learn from all the experience of software engineering exercises. In that you're going to have multiple people probably simultaneously working on your question bank. Those people will be working probably on different parts of it, but you want to be able to integrate that work without people stepping on each other's toes. Behind the scenes, there's for people to know what it is, a Git repository which manages the question bank. People can develop some code, other people can check it, and only when everybody's happy does it actually get committed into the main part of the repo. Another key thing is that it uses standard technology, so it uses HTML5 as the display language. It uses Python on the back end. These are common systems, and in fact the use of HTML5 allows us to take things that were on the web and turn them into PrairieLearn questions. If somebody built some neat little widget in your domain, you can usually take that if it's open source and make PrairieLearn questions from it, and I'll show you an example of that. The PrairieLearn doesn't have the notion of question types. If you go to Canvas, it's like, do you want to do a multiple choice question, or do you want to do a fill in the blank or an essay question, whatever? There's no notion of question types, but there's things that people do as part of questions that they do over and over again. For example, you may ask for a numeric input or a symbolic input or such and such, and so when we identify there's these things that people are doing over and over again, we can encapsulate them in what we call elements, and so that embeds all the codes so that somebody can use that code without having to look at it, and so we're building up a more and more sophisticated collection of elements all the time that let people, even without understanding how those elements work build more and more sophisticated questions. We're hiding a lot of complexity and making the tool still extensible. Also, for people in computer science, we have very sophisticated auto graders that are very flexible because you can configure a docker container anyway you want, so again, a lot of flexibility. Let me show you a couple examples just to point out the extent of the things you can do in PrairieLearn. By all means, this is not exhaustive. It's just what I cobbled together. This is my canonical example of a question generator. I teach a course on machine organization in computer science department. I'm showing a picture of a circuit, and I'm asking the students to give me a mathematical description of what the circuit does, and this is a question generator. If I say, hey, give me a new variant of this question, it gives me a new circuit, and these pictures aren't being stored, they're actually being generated dynamically. I have a piece of code that knows how to generate Boolean expressions, and I have another piece of code that given a Boolean expression knows how to draw it as a circuit, and so I can make as many of these as I want. Basically, again, the idea is that I'm going to give students as much practice with this idea as they need in order to master it. This is a similar thing in the mechanical engineering domain. This is from a dynamics class. There's two rods. It's giving the position and the velocity of the two points, P and Q at the end of the two rods, and it's asking what's the velocity of the Point A that's between them. This also is a question generator. All of these numbers are going to be randomized there, as well as this picture. If I say, give me a new variant, then it's going to give me a new configuration of the linkage, as well as the corresponding locations and velocities of those endpoints. Also, in our institution, we're trying to encourage our engineers to use programming as a means of solving engineering problems, and so we're providing the students all of the constants in this problem in a bunch of different languages. If they want to write a little code, to solve this problem, they can do that without making transcription errors, so I can just copy that, write some code, come back and solve that, and that's true both on homework and exams that we give them access to programming tools on exams. This is an example of a thermodynamics question. This is a little simpler that in this image, the image is static, but we're going to change some of the numbers in the image as we randomize the problem, so this one is 18 psi, and if I say, give me a new variant of this, now it's 24 psi inside there, and actually, it will change units. In some cases, it'll use metric units, so it's giving kilopascals instead., This is an example of what's called the order blocks element. Basically, I have a bunch of tiles that contain things, and the student's job is to order the tiles in a way that solves some problems, so this is what are called proof blocks. I can build a proof by moving these in some particular order, and if I grade it, it will tell me,

you got the first one wrong. Basically, it's giving feedback until it hits something wrong. In this case, I didn't know how to do that problem, so I got the first step wrong. This is a physics problem or, I guess, arguably just calculus, but it's plotting the acceleration as a function of time, and it's asking the student, given an initial velocity at times zero, such and such, what's the velocity at 12 seconds? I as a student, have to manually integrate this, so this is asking in numeric terms, and if I say, give me a new variant, it gives me a different plot of the accelerations. We can ask students to do symbolic math. There's a tool that either I can bring up a keyboard that lets them enter symbolic math or here's a lookup that, if I use LA Tech notation, I can just type it, and it will check, is your symbolic expression equivalent to the correct answer using SimPy. We have support for graphing questions, so we can, if I select that tool, I can drop down asymptotes. I'm not looking at the question, but saying mark all the intercepts on the axis and then roughly sketch the shape of this plot, and we can auto grade that, that will typically have very small tolerances for where the asymptotes are placed because even after you've placed them, you can reposition them, to make sure they're exactly in the right place, similarly with the points that maybe that point should have been over here. It's harder to free form, draw the line and get it in exactly the right place. We'll typically give the students a much wider tolerance, which is what we would do on paper, is that the calc faculty that we built this for would just have the students, like, label the positions of particular points and then just eyeball is this sketch, roughly the right shape. This is a questions from the robotics faculty. It turns out to be a multiple choice question, but what's interesting is that the answers are not numeric, they're theographical. It's saying, it's been a while since I took robotics, but what's the manipulability ellipse of the tip? When this is randomized, it will regenerate those multiple choice options. Those are being generated randomly based on the configuration of this problem.

Speaker 1: Are you doing that image based on the new L_1 and L_2, or is it just saying down below, there's a new L_1 and L_2, the work can be.

Craig Zilles: I think you could redraw it, I'm not sure this one does. Yeah, it looks like it's constant, but you could use that implementation to redraw it. Again, it's up to what the person writing the question did. The question was, is it redrawing this linkage of the L_1 and L_2? It certainly could, but I'm not sure it did in this question. This is a question asking students to draw free body diagrams, and so this tool lets you take and position where the forces would be. Again, it can auto grade this, probably shouldn't spend time doing this, but so for example, I might do something like that and save and grade this, and it says, in this case, this is what the answer should have been, so I got, I think, three of them, so it gave me 75%. But I didn't get the downward force of the thrust piece itself. This is an example of taking a piece of open source software and building a question around it. I'm asking them to design finite state machines for people who know what that is. That's great. If not, it's a certain kind of graph. I had found online somebody had built a nice little HTML tool to design finite state machines, and I extended it. As I'm building this, each time I do some action in this little CAD tool, I'm updating a string. When the student says, Submit, it's sending that string to the server, and so I can auto-grade it. If I build questions like this that capture the student's work in a digital format, I can auto-grade very sophisticated things without using AI. What I would call deterministically auto-graded. In this case, I haven't actually even finished it, and I can say, Hey, go ahead and grade it. This question will refuse to grade it because it says, Hey, things are incomplete. You didn't set the output state in this state, and then there's a whole bunch of edges that are missing. I, as a student on homework, get this feedback, and I can iteratively try it. Or on an exam, I can take that feedback, and as an instructor, I can give the students slightly less points each time they submit it until they get it correct. We have support for sophisticated code-writing problems. This is running VS Code inside of PRI Learn. If you want to do multi-file programming assignments, we can do that. This is support for image capture. For our calculus class. They wanted to be able to not just grade the final answers, but grade the derivations. We added support for image capture. On homework, we allow them to use their phones. In the testing facility, we've added webcams to a bunch of our computers, and so we can capture the image, and then we can say, Here's the part that I actually want to grade. That's not

really very interesting math, but. Our calc faculty use this for manual grading. That PL has manual grading support that's very much like GradeScope. You can see it here on the right. We've been doing a bunch of work with getting it to do AI grading. This is in the experiment that was run last fall of auto-grading handwritten work. We're randomizing the question. We built a rubric. In general, if you're building rubrics for AI, you need to be much more detailed than you would be for humans because the AI is bright, but it doesn't know what you want, so you have to be very clear about what you want. We'll take the original question, the student's work. Usually, we'll compute what the correct answer is for the given question along with the rubric, and we'll send that to the AI, and it will go through and say, Here's what I think the student wrote, and it will set all the rubric items, whether they should be granted points or not. There's actually two columns here. One is the manual grading column. One is the AI grading column. You can see here that the AI, is actually on this one, selected the rubric items similarly to the human. Typically, this is being done in bulk after the assignment has been submitted by everybody. The faculty member or their course staff will go through and hand-grade a bunch of elements, then run the AI against it to see, is the AI performing similar to the human. If not, they refine the rubric until it is. My understanding is that AI is actually very good at refining the rubric that you can say, Here's some questions that you graded differently than I did. How would you change the rubric to make sure that you grade them the same way that I did? Then you can hand-grade a couple more and see if it's on track. I'm a long-term. It's much more compelling to use AI grading for formative things because giving the immediate feedback to students is very valuable so they can know exactly what they did wrong and try again for exams. I think we can probably afford to still manually grade stuff for high-stakes assessments because I think students are willing to make the trade-off to get some uncertainty when the trade-off is getting immediate feedback. But they're maybe less excited about having AI grade them on high-stakes assessments. Other questions? There's another question back there.

Speaker 2: Regarding the variations that it auto-generates. Thanks, Eve. This is not my field of expertise. I'm looking at circuits like that or a logic diagram like that versus the variant. To me, that might be like one looks significantly simpler than the other, but then again, that's me not being in the field. My question is, Is there a knob that you can tune on how the level of difficulty, it's based on the question? I would assume that you want the level of difficulties somewhat similar between the variants?

Craig Zilles: This is all done by me. I was the one who decided these are an equivalence class. I'm not saying I'm perfect. There's a lot of people that are very concerned about perfect fairness. I am not one of them, and I think it's misguided that when we run tests. On homework, I'm not worried at all because, basically, they can repeat the task as many times as they want until they're satisfied they do the thing. If some of the variants are slightly easier than some of the other ones, who cares? Obviously the place where the fairness matters is on a high-stakes assessment where this is going to nontrivially impact their grade. A couple of things. One is, I try to avoid where there's any one question that's worth a substantial portion of their grade. Like, the one time I know of there was big fairness concerns was in a class where there was a two-hour exam that had exactly big coding problems on them that were each worth like 7.5% of their final grade. The students were very sensitized to which version of the question they got. One thing that is true is that the students are concerned about fairness, but they're not very good at actually knowing which versions are harder than others. Everybody thinks they got the hardest version, which obviously can't be true. The other thing is, so I'm the one who's judging, are these roughly equivalent? In practice, when I write, so when I write different versions of the problem and I put them in a question pool, I try to get them so that they're empirically within about 10% of each other. If my exam has 10 or 15 questions and they're all within 10%, then maybe some are slightly harder and some exams are slightly easier, but it's roughly the same thing, and the amount of variants that I'm introducing into their grade is much less than if one of my colleagues teaches the class in the next semester and has a completely different organization of the class, that class version, the grades are going to mean different things. The other

thing that I keep in mind is we teach this much stuff, and then we actually only test this much. There's a lot of sampling error that's going on that the student that happened to study the things that were on the exam versus the one who didn't. There's a bunch of measurement error already there. You're just trying to get the error that you're introducing through randomness to be in the same scale as the existing sort of measurement errors. I do my best that there is every once in a while, I'll get one version here that seems somewhat simpler. We did a study to try to look at this problem that we basically looked at all the different versions for a series of exams, and there were specific variants that we found that had in general, it was rare that there were specific variants that faculty had decided these were similarly performing and that were wildly different performing. Some examples of that are people were doing things on the unit circle and things in quadrant three, where X is minus and Y is minus, turned out to be harder than things in the other three quadrants, which, in hindsight, makes sense. Sometimes if you're varying something from a negative number to a positive number, that zero is much easier or much harder than all the other numbers. Anywhere there's a discontinuity seems to be a place you should be concerned. But we were pleasantly surprised that we were taking random faculty and just looking at what questions that they had written that, for the most part, the randomness wasn't. The faculty's judgment of how to randomize the question was pretty good. That their pedagogical content knowledge about what's hard and what's easy was a pretty good guide as they designed the questions. There's another question here if you want to throw the microphone.

Speaker 3: Did they have to do this question or not? Because I say just Z and not Z. I don't know, it's been a while since I looked at those circuits, but I think it's Z and not Z, which is obviously zero, and then it's going into an OR. Basically, this one ends up being not X and not Y or something like that. But if they had to simplify and some of the other ones didn't simplify so easily, then that would also introduce.

Craig Zilles: It's funny. It's Z and not Z.

Speaker 3: It has Z and not Z, and then it's going into an OR with something else. Eventually Y came out of that hole.

Craig Zilles: You don't need to give it a minimal version. Just being dumb. But I think you're saying this term actually isn't necessary, so it should be sufficient.

Speaker 3: The next one, too, because it's basically a zero going into an OR. In the end, that whole thing comes out to be not X.

Craig Zilles: What this question is doing behind the scenes, the expressions aren't unique. There's multiple correct expressions. What I'm doing is I'm generating a truth table from the expression and then checking the truth table because the truth table is unique.

Speaker 4: Can you show me?

Craig Zilles: I can show you other questions too that may be more interesting. But there's three main parts to a PrairieLearn question. One is the metadata associated with the question. This is just like what the name of the question is, maybe some tags about what the topic is, and whatever you want to store. In this case, it's a circuit and expression question. If I want to later search for circuit questions, I can do that. There's the presentation layer. This is what determines what students see. This is in sort of extended HTML. Anybody who's familiar with HTML will recognize this is wrapped in a paragraph tag. There's some special tags. This is an example of an element. Here I'm saying this is the Prairie string input element. I want the users to type something, and virtue of a string is that I'm not going to constrain it to anything. As compared to an integer input element, that would only allow them to type numbers. They couldn't type any punctuation or any letters in an integer input element. String input element, they can type whatever, and I'm just going to deal with whatever they type. There's also a figure element that's saying, Hey, I'm going to draw a picture. In particular, that picture

is being drawn dynamically. Then there's also this, so if I go back to the question, go back to the view that I had before. There's actually three ways that you can view a question. This is what we would call the question panel. This is what the student gets shown that they can interact with initially to answer the question. There's what's called the submission panel, which is every time I submit, I'm going to record what the student did, and I'm going to display them so they can go back and they can see their whole history of attempts on this version of the question. Then the third is the correct answer panel. This is what the student sees when they can no longer attempt the question. On an exam, if they can attempt a question four times, then after the fourth time, we'll show them whatever you decide to render in the correct answer panel, or as soon as they get it right, they get shown the correct answer panel. On homework, we often give them exactly one attempt. You can try this version of the problem once. If you didn't get it right, we show you the correct answer, and then you can ask for a new variant of the problem. PrairieLearn uses a don't repeat yourself concept, which basically, we're going to generate all three of those views from one file. In this file, you demarcate which things should only show up in the question panel. This is saying for the answer panel and the submission panel, don't show this stuff because it would just be extra garbage on the page. That's what you should show on all of the panels is this string input thing. This input element gets rendered in three different ways in the different panels that in the question panel, it lets you enter a new answer. In the submission panel, it shows you what answer you submitted previously. On the correct answer panel, it shows the correct answer. Then there's stuff that we only show in the answer panel, which is like, I'm giving you some feedback of this is what the correct answer is. I guess this is one solution is correct answer. You can see here this is what's called a template. This is something that's being computed by a piece of code that's being jammed into this hole. This is a hole in the document that we're going to fill with something that's being generated by running code, similar to this feedback message. I also have a link to a video. If you want to see an example of me solving one instance of this problem, you can collect this video, and it won't be the instance that you got, but hopefully it'll give you the process so that when you do it the next time, you can get it right. The last piece is the logic of the question. This is maybe one of the lesser impressive things, but whenever a student asks for a new version of the question, we'll call the generate function if you choose to implement it. There's a bunch of functions you can choose to implement or not implement. In this case, the generate function gets called, and I have a piece of code that's in a library that I wrote. I have a bunch of questions like this. I wrote the code once, and I put it in a library. This library says, Call this function that says, Give me an expression that has three inputs and is of difficulty one. The library that I wrote there's a bunch of if I said difficulty zero, I would have gotten some simpler expressions than the ones here. It's generating that expression, and then it's putting it into the database. We pass in this data object. Not all. If you're overwhelmed by this, I would say don't be because a lot of even when I write these questions, I do a lot of cut-and-paste coding where I take a question that is very much like what I want to do, make a copy of it, and only edit the things that I want to change. I never remember what the right format of this is. But basically, we're taking this expression I generated, and that's going to be the correct answer. I'm sticking it in this data object, which gets stuck in a database by PrairieLearn. The correct answer, and apparently, for some reason, I decided to hardcode this thing here. When the HTML goes, and it tries to load this figure because it's a type dynamic. It's going to call my code. It's going to ask for a file, and I say, if the file you're asking for is this, what I'm going to do is I'm going to reach back into that data object. I'm going to get the expression I generated, and I'm going to call a function that knows how to draw that, and I'm going to send that picture that I generated back. It's three input functions, the generated circuit. I was lazy, and so I passed that information. The last piece is the grading. This is a question where there's multiple possible correct answers. If there was one correct answer, I could compute that correct answer in this generate function, and I'll show you some examples of that. I tell the element, here's the correct answer, check to make sure they gave the correct answer. A numeric input element would have a tolerance and check to say, is their answer within some tolerance of the answer that you said was correct? If it was symbolic, we would say, can I manipulate this to get an equivalent expression. In this case, there wasn't a built-in can you

manipulate this Boolean expression thing? What I do is I take the data object that Plen hands me, which now contains the student-submitted answer for that expression, and I try to generate a truth table from that. But the student may have written gobbledygook. If they did, this function is going to fail, it's going to throw an exception, and in which case, I give them a zero, and I say, your submitted answer is garbage. If that doesn't happen, then I take my correct answer, and I generate a truth table for it, and I say, if their truth table is the same as my truth table, they get 100%. Otherwise, they give a zero. By the way, here's some other information. I guess the feedback message is zero in this case.

Speaker 5: The professor or the TAs or whatever have to do all of this for every question?

Craig Zilles: Somebody or something has to do this. Let me shift gears a little bit. Do you have another question about [inaudible 00:37:01] question? That gives you hopefully some idea of what the internals look like. Let me shift gears a second, and talk about writing questions. Best practices for when you're writing computer-based questions. Implicit in a bunch of this is that we're going to try to auto-grade as much as possible. I'm not an auto-grading zealot. I think you should autograde everything that you can autograde, but you shouldn't limit yourself to only asking questions that can be auto-graded. But there's a big bang for the buck. You get this fast feedback loop for the students for things that are auto-graded. Things that make auto-grading easier. First is make each question independent. Try to avoid things where you have seven parts to a question, and Part B depends on Part A, and Part C depends on Part B. Try to isolate learning objectives. If you have a question that's testing, do they know three different things? If they know two of them, but they don't know the third, it's hard to correctly assign partial credit deterministically. That's a good thing for trying to use AI if you want to. Another recommendation is to try to minimize the amount of storytelling. People often have these elaborate scenarios where they're like, blah, blah, blah. Two things. One is, if you're going to have many isolated questions that waste the student's time, and I think it's also an equity issue that if you have non-native English speakers taking your class, that storytelling is going to slow them down, and it may have references that they don't know. If you want to know, can you do this thing? Ask them to do the thing. The question was, do your instructors, and TAs have to write these questions? Five years ago, the answer was yes. The answer now is maybe. I'm a huge advocate of having AI do boring stuff that I don't want to do, and it's good at writing questions, my mental model for AI is it's a middle of the pack TA in that, my experience is my best TA is, I can say, go write a question on X, and they write a beautiful question. I'm like, no notes. Let's use it. The middle-of-the pack TA can do some things well, but they can do other things not so well. Ideation of I like, I tend to write the same problems over and over again. I like getting ideas from my core staff, like, what questions do we write? The ideas are sometimes good enough to help me think of questions I should write, but rarely would a middle-of-the-pack TA come up with a fully formed question that I could use out of the box. AI is good at coming up with ideas. As far as posing questions, sometimes, it doesn't ask it the right way. There's a lot of pedagogical content knowledge. If you've taught a class a bunch of times, you know how to strip out all the stuff you don't care about and isolate the thing that you care about. Middle-of-the-pack TAs aren't very good at that. They are very good at implementing questions. If you tell them, if you spec out the question, they can mechanically generate the question. AI is very good at that. I'm also very worried about the AI's ability to randomize, and generate similar difficulty versions. I think there's a lot of pedagogical content knowledge that we as instructors have that indicates what things would make a question harder or easier or the same difficulty?

Speaker 6: Can I ask a quick question?

Craig Zilles: Yeah.

Speaker 6: This came up several slides ago. But you said, check the answer and make sure it's symbolically equivalent to the answer that you give. The question often is, what prevents the student from copying the question, and putting it in for the answer, and it seemed, no, wait. I wanted you to simplify that expression.

Craig Zilles: That's not in the hands of the students. Basically, they type in their expression, and then that gets sent to the server. On the server, we're checking is their version equivalent to what we said was the correct answer.

Speaker 6: The question is equivalent to that [inaudible 00:41:29]

Craig Zilles: I haven't taught a math class, so that's an interesting question. In this case, this is an example of a non-math person asking it to write a calculus question. Many people use AI to write PrairieLearn questions outside of PrairieLearn. We have added support insight of PrairieLearn for people that don't want to do that, be like a one-stop shop to write questions using AI. The prompt at the top right is what I told it. I said, generate a calculus question where students have to evaluate a definite integral. The integration should be over X, but the question should also include terms involving Y so that they have to provide a symbolic answer. Maybe one of you could tell me what the correct answer is. We could test it, but it thought a while it generated this.

Speaker 6: My concern is if you write in that whitespace, integral [inaudible 00:42:39], then it would say, yes, that's correct.

Craig Zilles: Certainly, you could do things where you could be like scan the answer to see if there's any instances of X there, and if there are, give it a zero write off.

Speaker 6: Or if it had an integral sign, and the answer.

Speaker 7: There are ways to fix it, and other ways to cheat too. There's ways to fool AI.

Craig Zilles: This is an AI, this is being done deterministically. There's a Python library called SimPy, which does symbolic math manipulations. I don't know the specifics of it, but my guess is so, yes, you do have to be careful about students abusing auto graders that typically, when we write coding questions, what we'll do is, so I'm going to say save grade, we'll often have a couple of hard-coded test cases where, the students can see, on a particular input, did they get the right answer. If you only do these, then they can hard code your function to like, if they submit one, you should give this answer. If they submit seven, you should give this answer. we'll typically give most of the points. I would actually have 70% of the points being done by random test cases that change every time, and that the students can't see. You do have to think about, what are the ways that students might abuse my system? But I'm pretty confident that we can we can deal with that, you can review the students' work, and see stuff. Does anybody want to give me a prompt, either come up and type something or try to give a somewhat what?

Speaker 7: Can I ask a different question? One of the challenges that one faces if one wants to go to PrairieLearn is onboarding an existing class, the writing the questions is non trivial. could you talk about the process that one would go through over maybe multiple semesters or quarters of porting an existing class into PrairieLearn, and how that plays out, and why it's maybe not as bad as it sounds?

Craig Zilles: The first thing that I would do, if I had never used PrairieLearn before, and I was teaching a new class in, I would go join our Slack community, and I would go join the general channel, and I would introduce myself and say, "Hi, I'm teaching X for the first time using PrairieLearn. Does anybody have any questions that they could share with me?" I've had a lot of experience. I gave Leo Porter a bunch of my CS1 questions that he used in the CS1 that he revised, and my machine organization questions that they've, I believe, used here. Our experiences have been that when people write question generators, they're not so concerned about their questions going out because what makes my exams trustable isn't that I'm keeping the exam secret. That's not the primary thing. It's that there's so much randomization in the exam that you can't fit all the stuff in your head without actually learning the material. People are very willing to share stuff. I think a second important thing is you don't need to shift all of your stuff to PrairieLearn all in one semester that for many years, we didn't

know how to do our cash exam as well as we wanted on computers. We had six computerized exams, and one paper exam. you can piecemeal things. A mix of auto grading, and manual grading, I think, is also useful in our data structures class. The last question on all their exams is, here's a situation? What data structure would you use to solve this situation, and why? The other thing is basically you start off with the easy stuff. You can make some hard-coded questions. You can make some questions where here's a picture. I'm going to randomize the numbers. You do the easiest stuff first. What takes a lot of time is doing the first question of a given type. For example, the first time I did one of these finite state machine questions, it took me like two days to get the first one working because I had to get all the pieces working. Once I had one working, then I could bang out more of them in half an hour or whatnot. The big advantage is rather than rewriting all of your stuff every semester, you can reuse the stuff you used last semester. You don't have to rewrite that because if you've written it up as question generators, you can extend your question pool, make it bigger, and bigger every semester because that effort you would have spent by rewriting everything you could instead put to writing more stuff. I don't know, Steve, is that.

Speaker 8: Is there any checking when people go onto the board to make sure they're not a student instead of a faculty or, can you imagine I'm signed up for your class, and I like, go on board and say, "I'm teaching a class on whatever it is, can you send me the question?"

Craig Zilles: So far, I've shared it with people that I have a personal relationship with. But there's a lot of, I'm also not super concerned about that. Again, my mental model is, if my question pool is large enough, I could put it in the public domain, and it wouldn't matter because it's hard to memorize a bunch of stuff that doesn't mean anything to you, that I'm not trying to stop students from cheating. I'm trying to make it not profitable, that I want the rational strategy in my class to learn the material because that's the easiest way to get the grade. There are students that will still try to memorize all sorts of stuff. They'll spend way too many hours, and it's not as successful as if they sat down and, learned the stuff. We see people trying to sneak notes into our testing facility of questions that were on the practice exams that aren't actually on the exams. It's like, what are you even doing? But I understand your concern. I think people would probably do a web search for you before they shared the stuff. But in practice, I'm not actually that worried about students getting my stuff because they're going to have to demonstrate that they can do the thing in our testing facility when they don't have access to AI or, communicate with friends that can look stuff up for them.

Speaker 7: I would add that one of the things that we do, also many of the professors in our department do is our tests are questions that they've already seen in the homeworks. We will have a massive testing bank with many questions, and they'll have access to all of them for homeworks, maybe two or 300 questions. then the homework will say that the exam is going to be a subset of these..

Craig Zilles: They don't know which?

Speaker 7: They're going to be randomized, and so go practice the homeworks. If you can do them all, then you're done. It also means your exams no longer have to be hard, you can make your homeworks as hard as you want the students to be able to do, and then verify that they can do that in a relatively straightforward way in the exams.

Speaker 9: Even if they have all the questions, go, learn how to do all the questions. That would be great.

Speaker 10: How realistic is that for actual university learning when they get out in the world and they're like, I haven't seen this question before. I don't know what you did.

Craig Zilles: We will typically, again, the instances of these questions are probably not going to be ones they've seen before. For example, the circuit, they won't have seen that circuit before. For

programming questions, we will typically put slightly different programming questions on our exams than the ones that are on the practice exams and homeworks. Example, I teach a CS1, and so one of the things I ask the students to do is given a collection of things, give me a new collection of things that contains the subset that all fit this property. I can write many different versions of that question. I know what you're saying is transfer is the thing that we care deeply about, especially as academics. I think usually that's the thing that's distinguishing maybe a B level student from an A level student that.

Speaker 10: Even levels of knowledge. There's a regurgitation question, and there's like simple application. At the top, there's like actual level.

Craig Zilles: But be honest with yourself what fraction of your students do you expect to do actual novel problem solving? Those are the people that were typically saying you should go to graduate school when we meet them.

Speaker 9: I think also you can you can also do manual grading. You can draw pictures, you can upload the images. Those aren't auto grade. Those can be in a different exam. You're not just bound to doing auto grading only, and those questions you could not release.

Craig Zilles: We were talking at lunch that you could auto grade a bunch of stuff in the CBTF and then have a paper exam that everybody takes at the same time. If you have one version of the question that's asking them to do novel transfer that each semester you write that question completely from scratch. I think it doesn't have to be an all or nothing to get benefit from using this approach. Just quickly flip through the slides, but people can pause the recording and read the slides, or I can quickly in 10 minutes go over a bunch of best practices.

Speaker 9: There's a question in the chat. Hold on. I was just curious about if there were any key features of the question generating scheme that maybe you wanted to implement when creating PrairieLearn but that didn't end up working out.

Craig Zilles: PrairieLearn is a living platform. PrairieLearn is open source. If there's a feature that you want that it doesn't implement, you are encouraged to reach out to the maintainers. I encourage you to reach out to them before you write a bunch of code and submit it as a patch, but because that'll hopefully save you some time. But they definitely take patches from various people to add new features. We were talking at lunch, one of the features that isn't there yet, but seems to be more and more interesting is the ability for a question to know things about its context, that right now a pre-learning question doesn't know which student is doing it, and whether the student is doing it as homework or on exams. That second part is mostly good in that in general, you'd like to be able to write a question and then reuse that same question maybe on in class activities, on homework, on practice exams, and exams. But there's reasons why you may want to have slightly different versions of that question. rather than making two copies of it one for each context, if you could say, hey, what context am I in, that could be a useful feature. That might be a thing that comes to mind.

Speaker 11: You pointed that out with a nice line there about the similarity in the complexity of the exam problems, and you pointed out it's very important. I also think it's important. What do you do to make sure that different students get exams of the similar complexity?

Craig Zilles: Mostly at this point, I've been running the same exams for a bunch of semesters, and so I'm pretty comfortable that they're of similar difficulty. PrairieLearn has pretty good statistics about after you run an exam, if you have a question pool, it'll show you what the average correctness was for each question in that pool. If you have an outlier or it's much higher or lower scoring, typically, I would take that out of the pool. The next semester that I ran it, I would feel a little bad, and then I would go on with my life in the current semester that my world is that I will give the students two things. One is, I'm asking them many questions over the semester, like those little things even out.

The second thing that I do is I do second chance testing. They have two opportunities to do well on my exams. If they're not doing well, it's probably not because of the random difficulty variances, it's probably because they just don't know the material. I sleep very well. This is not something that I spend a lot of time worrying about. It's also not a thing that the students complain a lot about. Only when there's a small number of problems that are worth a lot of points and they don't have multiple attempts at them, that seems to be like the thing to not do.

Speaker 12: Is it like picking, like four questions out of your bank of 200, or is it like Question 1, pick something from this group that's of approximate same difficulty. Question 2, another group of approximates.

Craig Zilles: It's much the second of them. My exams will typically, I call them slots on the exam. Each slot is guided by a pool, and that pool usually is testing a particular learning objective at a particular difficulty. For example, if I go back to this circuit question, so there's a very simple instance of it. That's the one that I'm embarrassed by. I have a whole collection of questions that are testing the same learning objective, which is, can you take a Boolean thing in one format and convert it to another format? The formats I have are circuits, Boolean expressions, truth tables, and [inaudible 00:57:32] Can I give you one of those and you give me back another one. My question pool is, for a certain difficulty it's like whatever nine or 10 different versions of problems where I give you one version and ask for a different version.

Speaker 12: You'll have this one with different circuits, and then you'll have a separate problem that also has variations.

Craig Zilles: I give you a random truth table and ask you to write the Boolean expression, or I give you a random expression and ask you to fill in the truth table. I show you a circuit and ask you to fill in the truth table. I want you to be fluent with this notion of like, these are all representations for the same thing.

Speaker 12: If two students take it at different times, because you are saying that students can take them at different times, then the first one leaks that there's this question. The second one doesn't get that same exact question, but now they know they're going to get something that's still of the same.

Craig Zilles: I've already pre-leaked this. What Steve was saying, I will give them a practice exam generator that has this exact pool on it, that when I have a big pool of question generators, I don't keep it secret. I, in fact, want them to grind that pool of question generators and be able to do all of the things. I can be like, great, you can do all the things. I'm not teaching to the test is only bad if the test is bad. Once you have enough stuff you're just like, hey, go to town. Initially, when you make this transition at first, you may not have a lot of questions and you're worried about, we use the AI term of overfitting where it's like if your test set is the same as your train set, they're going to do disproportionately well on that. But once your test set is enormous, if your test set covers all the things that you care about, then you could train on the test set and be happy with the consequences.

Speaker 12: How long would it take somebody getting into PrairieLearn from scratch to get up to where they have a bank of 200 questions or whatever you need?

Craig Zilles: It depends a lot. A, if you know somebody who can give you 200 questions, then a few minutes. If there's already questions like what you want to do. I basically built a CS1 course in real time one semester. I've been using PrairieLearn for a long time in my machine organization class and a couple of other classes. I was teaching the CS1 for non majors for the first time. One fall semester, I'm like, over the summer, I'll write all the questions as we as faculty are want to do. By the end of the summer, I had two weeks of material, and I basically wrote the rest of the material as the semester was happening. In addition to producing all the lectures and whatnot, because I was completely revising the question. It was taking me as a relatively experienced pray learned person key thing is if

there's a question that's the same form as what you want to do, so that what's a good example? Like this thing where it's just like, there's going to be a picture. I'm going to randomize some numbers around it. I can make a copy of this, and in 15 minutes, I can produce the new picture, I can write the text, I can write the randomization, and then I send it to my TA to check a bunch of times to make sure it seems reasonably good, and I'm done. You can blast out these questions pretty quickly. What low is doing the first of a new question? At this point, there's a lot more things, like now you can do a grabbing question in relatively quickly or one of these Parsons question or order block questions where you just have to come up with the proof or the piece of code that you're going to turn into a Parsons problem. I guess you have to think about the randomization piece a little. But maybe you don't have to randomize everything initially. But it can be done by one person in a slightly unhappy semester in real time as a bound.

Speaker 9: I was more of a novice when I did this, and I PrairieLearned a course. I got some materials from another professor. I went through and did wrote translations of the exams and the homeworks and went through and just did the math once and then put in some random numbers to make it a little bit harder, and that took maybe like 15 minutes a question. Then I would go through my slides, and I wrote down all the vocabulary words and all the key concepts, and I handed a list to a TA. I said, go make questions about vocabulary words for these 10 terms and then we would go over them in our weekly meeting. That got most of the way in another quarter. Then by the third quarter in that quarter, I was being pretty careful about. I didn't have enough questions, so a big question bank and use them on the final. But then by the quarter after that, I had a big enough question bank that everything was pretty much running. It took, I guess, by the third iteration, things were in good shape.

Speaker 12: What situation with a lot of textbooks that offer question banks that instructors can use in online curriculums?

Speaker 9: I didn't have that situation in my class. There's not a great textbook for the class I was teaching. But I guess you could you could draw on those question banks as well or maybe look at them for inspiration.

Speaker 12: Do those question banks exist for any person?

Speaker 9: I don't know. Was anyone publishing PrairieLearn?

Craig Zilles: Not that I know of. Again, definitely what I would recommend is if you have colleagues that you find out that are teaching the thing that you are teaching that have used PrairieLearn, you should just ask them for their questions because many people are willing to share that one of the big thrusts now in PrairieLearn is to facilitate question sharing. The old way of sharing is I give you read access to my GitHub and you can just take whatever you want. Now support inside of PrairieLearn where you can mark questions saying, Hey, I'm willing to let other people use these. if you know about them, you can use other people's questions through the platform, which the advantage is that you're not forking the source. If they fix bugs, that you get the bug fixes. There's not a good discovery tool yet of what I want is the yelp where it's like, give me a question on Dijkstra's algorithm, and there's all these upvoted questions and I see, Steve Swanson wrote this one, so I know he's good, and so I'll take his question. That's the future that we want to get to, and then somebody then curates all of the best questions into here's a good assignment on such and such topic. We're not there yet, but I think it's just a small matter of people rolling up their sleeves and doing the work that I think again, people are willing to share stuff. It's just getting people together and putting stuff out there.

Speaker 11: May I ask how powerful is the AI generator? I don't know that we saw it in action.

Craig Zilles: This is still a work in progress. It seems pretty good at questions that involve a single line of input either like you can have multiple choice questions if you cared about those, but a single input

of a numeric question or some one line input symbolic or whatnot. If you want to try typing something, it doesn't do programming questions yet, at least the last time I test drove it, and it was struggling at doing visual stuff that at that time, the agent wasn't able to see the errors that were happening in the browser. It was drawing a bunch of stuff that didn't end up getting rendered because there were JavaScript errors. But the developers know that. But if you want to either type something or tell me what to type.

Speaker 11: The questions that you showed that you developed yourself and coded up. This AI cannot be used for such questions, is that right?

Craig Zilles: Right now. But if you were comfortable using something like Claude Code, though, and you had an example question, I'm very confident that you could get something like Claude Code to say, look at this directory to get the idea of the structure. Here's what I want you to do. Make a new question that does this using that structure. Here's the PrairieLearn documentation link, read the documentation if you need to. You haven't if you are somebody who's willing to do programming and you haven't played with Claude code or the like yet, it's a trip that it's quite impressive. That stuff will get integrated into this, but they are only rolling out stuff that they know works. Right now, they haven't set it up to have the examples of doing the programming questions.

Speaker 9: Any more questions online or in the room? Let's thank our speaker once more [Alarm]