Krish (products.snowpal.com) (00:01.162)
Hey there, hope you are doing well. In this video we are going to take a look at a feature, one feature, it's called relations.

And we're gonna I'm gonna show you a little bit of what that featured the actual functionality as we've implemented it on our Project Management platform we have it on the web and mobile apps. We're just going to take web app as an example But that's just one part of it what I want I would like for you to also understand if you're watching this or listening to it It's better to watch this video even though you can listen to it as audio as well I'm gonna because we're gonna do a screen share and actually show you a few things

is as a creator of applications, how this functionality, a singular functionality, which is handpicking one of our many features, a random one, to be honest. Just to show you how leveraging our API, and integrating slash integrating API, is gonna save you, your dev team, a ton of time as a company, a lot of money, and eliminate or at least mitigate a lot of your risks.

Instead of rebuilding or reinventing the wheel, should always leverage stable, scalable, extensible APIs that are out there. I don't know how many else that are out there. We are certainly there, Snowpal APIs. So today's feature is called Relations. Let me give you a quick introduction to the feature and then we look at it and then also kind of see how the API integration can work in this scenario.

Relations is, we call it relations. What it does really is, you have disparate pieces of content that you're creating. It's not very about what the content is, but it's some content. And we're gonna take some examples, which is gonna manufacture an example so we understand this. After I do a screen share, just a moment.

Krish (products.snowpal.com) (01:51.062)
When you want to connect disparate resources and want to be able to navigate and maneuver to them quickly, you can actually leverage the power of this piece of functionality that we call relations. Essentially you're relating disparate pieces of content. I call it disparate, but they could have something, you know, it is disparate from a content creation, content management standpoint, but they could mean something collectively for you. And if that's abstract, that's all right, because I did not,

it does sound abstract when I tell you that but when you see it in the moment it's going to clarify. Remember this before I do a screenshot that you're connecting content so it's easy for your users to navigate and go from one place to the other and I called it intentionally as disparate content because they may not have anything else to do with each other meaningfully other than the fact that the user is potentially working on three different items and they want to be able to quickly sort of maneuver and get

to the other. But that said, let me do a screen share. I'm going to share a window that

Our production system is on snowpal.com, the production web application. We have many products, it's one of our product. I'm just gonna, typically during these recordings, I use localhost, simply because it's just test data, and I don't wanna pollute our production data. Plus, if I log in as myself to our production accounts, it's got sensitive data, obviously, that I'm unable to share, as you can imagine. But again, when you're checking this out, just go to snowpal.com, and you'll see the exact same thing that I'm showing you here.

here. Let's go pick one of the, you know, I created some content, you know, it's a dev content. This is what key called courses. And there's a couple of blocks called math 101 and science

101 completely manufactured dev data, as you can tell. Let's add one more block here. Let's call it, I don't know piano.

Krish (products.snowpal.com) (03:51.01)
And maybe I'm just gonna add description saying start learning

learn how to play a piano.

Krish (products.snowpal.com) (04:10.642)
Okay, that's good enough. So I'm gonna go here. We had two blocks already. We had a third block called piano one. I mean, maybe even we can change this to one or two or something here. Now, let's say I'm gonna sign up for piano lessons as a user. I also have my math lessons, and maybe we can rename this to course tutoring or something like that. Let's say I'm just, I don't know,

manufacturing these words here.

I'm not teaching but I'm perhaps learning or maybe I'm a teacher who's teaching this. Yeah, it's in a teacher key. This is one of the type of keys that we actually support to have functionality that's specific to the world of education, so to speak. So let's say I'm a teacher, I'm teaching three different courses, math, science and piano. Now in the real world, perhaps it's unlikely that the same person is teaching three different courses but it's all right. I mean, I can imagine a teacher who knows that and also plays an instrument.

So I'm going to set up my courses. I have to go here. I have to create assignments and quizzes and whatnot. I'm trying to structure my content as a teacher. Let me make sure I'm still recording. Yep.

As I do this, I want to be able to get from one place to another quickly. Now, again, math and science, I have a different set of students. Probably there's no overlap. So they are disparate as we use that language earlier, disparate pieces of content. But now as a teacher who's starting the semester, I might say, hey, you know, I'm going to be teaching three different courses or tutoring three different courses. Course tutoring probably is better language, not courses tutoring. Not that it matters for what I'm going to say here, but it's just noticed. And it seems.

Krish (products.snowpal.com) (05:55.504)
bother me, okay. Let me go back here. So I'm gonna be like, okay, I'm gonna spend the next day or so trying to structure this content, add assignments, create my handouts and all that kind of stuff. So I'm gonna be working on them together at the same time more or less. I wanna actually, when I'm here, I wanna be able to get to the other two courses. Now you could say that this happens to be in the same sort of structure, so maybe that level of navigation and maneuverability is perhaps useful but not absolutely necessary.

because it's still easy enough to get from one to the other by coming here. So I'm gonna make this a little bit more, let me create another teacher key called music tutoring. Let me go into course tutoring. We actually wanna move this. Let's move piano from where it is right now to actually music. We move it there.

So we have two different keys here, course and music tutoring. And then under music, perhaps let's add one more. Guitar 30.

Forward is an advanced guitar class that I'm teaching. Okay, now if I go to the main page, the key listing page, and then if I go to course shooting, I'm like, okay, let me connect Math 101.

Let me go to Math 101. I'm gonna hit relation here. I'm gonna say, it's already connected to signs, as you can see, but I also want to connect it to guitar.

and maybe also want to connect it to piano because I'm going to be working on these three at the same time. So I did that. Now I come back the next day, I log out, I come back and I go to music tutoring, I go to blocks, I go to piano 102 and I'm like, okay, I'm making some changes, blah, blah. Maybe add a comment here.

Krish (products.snowpal.com) (07:58.158)
to add a collaborator or something of that nature.

Okay, and then I'm like, okay, now I wanna be able to go to the other courses that I'm currently working on. I go to Relations here. I see that Math 101 is actually connected to Piano 101. So I can go navigate and find myself there. So essentially, I was here, hang on. Yep, I was here. So let me, this takes me to the system keys. So if I go back, and I go to Math.

and I go to relations. So math is connected to science, guitar and piano. So the relationship is bi-directional. If A is connected to B, then B is connected to A, but each of them can be connected to their own set of resources. So I actually, while I was writing explanations, I forgot what exact relationship we created here. So let me go to science and I'm gonna add guitar. Okay. Now I'm gonna go back here, go to guitar.

Now if I go to guitar and then I click relations, I find science and math. Now I can underlay it because guitar has nothing to do with math, I'm gonna underlay it. So that connection has been removed. So this is literally for navigational purposes and we apply the same thing to our mobile apps as well. So obviously when you connect to resources and relate them here, they are related on all clients obviously. This is an example of a piece of functionality. We've used it in a certain way

Now you could take the same piece of functionality and let's go to one of our APIs. Let's go to building blocks. And then let's say go to blocks.

Krish (products.snowpal.com) (09:42.126)
I'm trying to see whether we have relations categorized, oh, we do have it here. So you go to relations in this collection, and then you go look to see get relations for a key, get relations for a block, relate a key to a key, unrelate a block from a key. So you have all of these permutations and combinations. Now we have three pieces of content, three levels in our content hierarchy by default. You can make it infinite when you use our APIs, our web implementation keeps it at three levels.

blocks and pods. Now you can relate a key to a block, block to a pod, and in all directions. That's what this, the list of endpoints that we're looking at here, that is what they let you do. Now you could take this endpoint and implement something quite different from the way we've shown here. You could take relations and, you know, depending on your use case, whether you're building software for the fintech industry or for the pharmaceutical industry or for, I don't know, the education industry or some other industry,

you could actually take this and see what you want to be able to connect or relate and then use it that way. You might expose this obviously not as relations as something else entirely to the end user but your backend team or the team the frontend team that's leveraging our APIs you actually don't even need a backend if you use our APIs. They are going to call this endpoint and they're going to say slash pods id relations something. That's just our language, our verbiage, our

makes the most sense to you. You just call this and when you expose this to your end users, whether you're building b2b or b2c apps mobile or web or microservices, you'll consume that and create your own functionality. Now imagine if you had to implement the APIs for relations to relate, unrelate, to search the whole nine yards. It's going to take you a whole lot longer than just to build a UI. The UI as you can tell here it's quite we've kept it as simple as possible.

but it achieves the purpose that we want for it to achieve. The API on the other side, the endpoints is a large number of them because nothing is easy. Yet, given the fact that we at Snowpal do full stack polyglot development, it's our opinion that building server scalable, extensible, robust server-side implementation is harder than building clients, simply native like web apps, the web or the mobile interfaces, because of the number of things

Krish (products.snowpal.com) (12:11.792)
actually worry in terms of integrations, validations, data, not just data integrity, but just the integrity of the system as such, security, everything else you have to generally worry and also making it very generic and fundamental so different businesses can create different solutions using the same set of APIs and endpoints. So implementing these endpoints, that they run into a number of them, there's a large number of these endpoints. I'm just scratching the surface here when it comes to relations.

is gonna be time that you save by not implementing these API on the end endpoints and by simply leveraging it. I'm not gonna show you the mobile version of it because it's just our mobile app that also supports rich project management functionality and one of the features is relations. But the idea here is to just show you how you could actually use relations. Take our production implementation as an example.

and then implement map it to your requirements. Just leverage, do the integration, purchase the API key and then connect to our systems in like 15 minutes and your UI team can start developing. Or if you wanna provision it in your own infrastructure, we are happy to do that. There's like five different ways you can license the APIs. Ultimately, it saves you time, money and effort and risk, reduces risk, eliminates it,

happy and I'm sure you're going to be happy as well. Thank you.