Krish (products.snowpal.com) (00:01.006)
Hey there, hope you're doing well. In this video, we'll actually go through the process of doing a simple integration with Snowpal APIs. And we'll not do any implementation work, at least not in this one. But let's say you are beginning to build something, whether it's a brand new web app or a mobile app or a microservice, or you're actually adding an enhancement or a feature, say a feature to an existing application. You may not...

may or may not have a backend team. Let's say you're a UI shop or you are engineers or predominantly UI developers and you have a UI team and you rely on other vendors or managed services to build your backend systems. Maybe you have something live in production, you wanna make some extensions to it. Or you're a small enough startup where you have mobile developers and you're just trying to build a mobile app to publish on the app or play stores and you...

don't have the expertise necessary expertise to build these backend systems. Or you are a big enough company or a mid -sized company that has these teams but they're not available to building this and plus even if they are available in another case, it probably is gonna take you a lot longer because those APIs are not readily available to you. The team is but not the actual functional piece of code. In any of these cases, you you're gonna be far better served.

and you can reduce the time to market dramatically by leveraging APIs. That's why we have so many APIs and thousands of endpoints. But how would you go about doing it? So let's take, you know, let's just take a simple example. Maybe it's a manufactured one, doesn't matter, and see what, how I would do it if I were actually not part of Snowpal and I was trying to build something for somebody else and actually wanted to integrate a scalable, extensible, available.

time tested API so I can be more efficient. That's the idea. So without further ado, let me share my screen and then we'll try to, let's see, okay. So we'll now go into the APIs. These are the APIs, but maybe we'll touch upon some of them, maybe, maybe not, but let's talk about API integration. So let's, as is customary, I'm gonna draw vertical rectangle.

Krish (products.snowpal.com) (02:19.118)
It's usually my starting point if you've seen my other videos.

I'm gonna call it Snowpal API integration

I could have picked a different type of diagram for this but that's alright. Okay, so let's try to be as creative as I can be.

Krish (products.snowpal.com) (02:50.872)
your product team comes and tells you hey dev folks build something for me and instead of calling it something whatever let's say that something is

Krish (products.snowpal.com) (03:07.084)
is a

Krish (products.snowpal.com) (03:12.302)
a new feature.

or a new application doesn't matter. So somebody's, you're starting to build something. What is it? There's not etched in stone, depending on the team, depending on what you're doing, you might take a very conventional or not so conventional approach. So we're gonna make a number of assumptions here. Say the first thing I wanna do here is ask, get my requirements clarified.

Krish (products.snowpal.com) (03:49.454)
Understand.

Krish (products.snowpal.com) (04:07.118)
Okay.

Krish (products.snowpal.com) (04:13.516)
The tool actually has this issue with sometimes the text being too tiny but hopefully you can see it. So the first thing you do is you understand the requirements. You work with your product team, essentially your product and...

product plus dev teams are gonna be working closely to ironing out these requirements. So once those requirements are ironed out, your dev team is gonna be like, okay, I need to start building. Let's presume that this feature that we are talking about here has this feature requires UI and API changes.

Basically front -end and back -end changes is what I'm trying to say here, right? Maybe that's a more generic word to write. Front -end and back -end changes. So that's what this feature entails. So your product teams, you'd work closely. Product and dev teams, and they let you know the requirements are good.

you're gonna start actually working towards implementing it, right? Not implementing, you're gonna start designing it, for instance. So when you get to the design phase, I'm gonna take a bunch of, use just a bunch of random shapes here to call it for some differences. They don't necessarily mean anything in particular, the shapes. Okay, design work.

Krish (products.snowpal.com) (05:42.102)
See.

I mean, since it's a new feature, there's probably not a whole lot of, I mean, depending on the feature, you may have some architectural work as well. If it's a new app, obviously you do. I'm just using it simply and loosely and trivializing a lot of these things just to get to the essence of what I'm trying to explain here. And hopefully I'm still recording this. And you need to do the design work. Now, if you, the design work is, there's two paths to this. So let's actually say,

Krish (products.snowpal.com) (06:16.11)
and I'm just gonna call, use the same language, front end and back end.

Krish (products.snowpal.com) (06:29.966)
Maybe the backend I can use, I'm gonna use like a dotted line here and I'll explain why. The frontend work, let's say you have to do. Now again, we have pre -packaged and we have mobile apps and web apps available. Oops.

Krish (products.snowpal.com) (06:52.27)
and the two.

Krish (products.snowpal.com) (07:02.122)
You can also, let me do this again, that's looks like an Invision bug.

So we also, you know, we have mobile apps on the app and play stores and a web app. You can go to like snowpal .com to check out our web app and go to ios .snowpal .com and android .snowpal .com to check out our mobile apps, native mobile apps on the app and play stores. So you can work with this to actually get started with the front end really quickly as well.

You have fully functional apps that can serve as templates. You can purchase licenses and you can use them. But since I'm going to, we're going to focus on API integration in this video, I'm just going to not dig deeper into that aspect. Let's assume,

you're going to be building the front end, making the changes, working with the React teams or Flutter teams or Xamarin teams or React Native teams, whatever your stack happens to be. But let's get to backend. Now, let's assume that for the functionality that you need to build, the front end team needs to actually work with the backend team because they need, I don't know, they need like, let's keep it simple, six endpoints, because it's not that big of a feature.

Now, now the API team can, the backend team can build these endpoints, define as, create a specification till the implementation is done. The front end team can use the stubbed out endpoints and work in that fashion, which works, but it's got its pros and cons. And your backend team is gonna, it's gonna take them time to do doing this implementation. Now your end user, your product team is like, hey, give this to me as quickly as I can. I don't wanna wait too much time because I made a commitment to the customer.

the customer is somewhere out here who's using your system. So you want to do it as quickly as you can and you don't want to reinvent the wheel either. What is not there, build it. What is there, don't waste your time, energy, effort and money rebuilding it. Use products that have stood the test of time and companies that are doing this for a living. That's like us. So you're like, okay, can I use someone else's backend system? And how do you make this determination? Here's where...

Krish (products.snowpal.com) (09:01.902)
You need to value your design and thinking head slightly differently as well. Sometimes as an engineer, as a developer, if you ask me to do six endpoints, it's depending on how much experience you have and what you've done in the past. You could churn this out sometimes fairly quickly. Take that with a grain of salt. There's a process involved. There's testing, there's regression, there's maintenance, there's extensibility, there's your experience per se, how much of this you've done in the past.

what the quality of your code is gonna look like and a number of other things. But a lot of us wanna just might think, why not I just build it? Because it seems like the most obvious thing to do, but may not be the best thing to do because it's already available. So why are you rebuilding it? Could you be doing, it's like, do not, I've said this before, but do not reinvent the wheel. We don't at Snowpal, we only build what's unique and where we adding value. Otherwise we look to other providers like open source libraries and whatnot to see.

what we can leverage to make our systems better and more powerful as opposed to just doing the same thing again, but differently, which makes no sense. So we don't do that. And I don't think you should either. So even if it's six end points, if it's 60 end points, it changes the story. It makes it even easier. So just by making it harder on our, on myself here on the Snowpal team to make this impression on you, even though it's only six end points, you still do not want to build it. Even if that's your first natural instinct. You want to go look to see,

Can I actually average something else that someone else has available? Now we have a number of these APIs. We have different videos that talk to the details of these APIs. I won't go into those details, but let me just say one thing. Let's say you started the building blocks API and let's pretend that your feature that you're building here has something to do with maybe restaurants. This new feature is maybe a restaurant feature.

For rest.

Krish (products.snowpal.com) (10:57.454)
So you might first look and see, you know what, does Snowpile have an API for restaurants? Let's start with the building blocks. There's no use of the word restaurants here, so the first thought could be, maybe not. Surely you're not gonna think that way, I'm just being preposterous here. What I'm trying to say here in a long -winded fashion is you have to just do the mapping between our terminologies and your terminologies. A key is a project, a view, a board, however you look at it. A block is something that lives within a key.

and a pod lives within a block and there's multiple resources in our content hierarchy. And you can have infinite levels of content hierarchy as well, our API support that. So even though we don't use the word restaurants, creating a list of keys is just language. It says slash keys. The actual endpoint says, if you go to the end points, add a key. It's gonna say slash keys. But you're creating, you're doing something for restaurants, maybe adding some feature for adding, creating new restaurants.

So all that you have to do here is essentially do the mapping between our terminology. Let me actually pick.

Krish (products.snowpal.com) (12:11.694)
you're gonna do the mapping between the Snowpal API terminologies. Let me put it here.

mapping between your language, your business language if you will and snowpal's.

Krish (products.snowpal.com) (12:42.126)
product terminologies. Now, if you need help with this, we can certainly help you out, but I think you will be able to figure this out really quickly by yourself as well. But we do offer professional services on aws .snowpal .com, where you can pick one of the professional services options and have us help you onboard and get started as well. It's entirely up to you. So the mapping between your language and terminologies, the first time it might take you a little bit of time and then once you get used to terminologies,

They are pretty standard English terms. They means, you know, not all of them are unique. There's some that are product specific. The rest of it is quite generic attachments, checklists, comments, notes and tasks. You know, they are what they say they are. Now we have something called a generic checklist functionality. You could have a user, you know, go pick a list of items that they add to your order or something along those lines. You might decide that you can use the checklist functionality.

to implement whatever it is that you need to do. So there's two paths to it. One is mapping of terminologies, the actual language itself, and then the next thing, actually I'm gonna say terminology.

Krish (products.snowpal.com) (14:01.582)
Sometimes I've seen that this tool does not.

Krish (products.snowpal.com) (14:15.822)
terminology.

Krish (products.snowpal.com) (14:21.902)
That's the first thing. Let's actually group this together, copy paste it. The second thing that you'll need to map is the, I would say maybe I don't want call it a component mapping. I'm just going to loosely call it functionality mapping. Hey, what Snowpal calls checklist is going to translate itself to an order, a list of items are you adding, you you're customizing a purchase at, at Kawa for instance.

In your order, you're gonna add, I want avocados, I want tomatoes, and I want something else that makes it into that order, but that's a list of items, and you could use a checklist feature that we have to present that differently to your customers as adding, you know, customizing an order, a food order. So that's the functionality mapping that you need to do. Terminology mapping, and then functionality mapping. I'm just gonna, since we took a random example, I'm gonna say checklist map to...

food order or something.

Krish (products.snowpal.com) (15:26.926)
So that's what you do. You say that's a mapping. Once you do that, then you essentially don't have to build anything. You saved all the time of design and architecting, designing, implementing, testing, deploying, upgrading, maintaining, extending in future, and all of those complexities are gone. You just do this and start consuming it. Again, how you consume this, we've talked about in different videos, the licensing.

slash pricing models as like seven different ways you can choose to integrate. I won't go into those details here just to keep the focus on the item that we start out to discuss. So that's that you start with that you do the mapping terminology and functionality mapping. Once you do those two mappings, what's left for you is basically starting the implementation work. So in the interest of completeness.

Krish (products.snowpal.com) (16:23.086)
implementation.

Krish (products.snowpal.com) (16:32.846)
Okay, so that's kind of how your approach is gonna look like. Now along this way, in the process, you might need some help so you're able to reach out to us in different capacities that we can work with you. If you want us to build the entire feature using our products as works made for hire, we are happy to do that as well. Our Manage Services team can make that a reality for you.

But the essence of what I've tried to capture here is do not end up implementing, don't make the decision to implement these six endpoints or 60 endpoints or 600 endpoints, whatever those are, unless you really, really feel that there is value added in you doing that. And for some reason, you don't see that our APIs don't provide that for you. Even if they don't, given that we have so many APIs and so many endpoints, adding something that you don't have is gonna be a whole lot faster for us, to be honest, than it is gonna be for you to build it yourself.

So that's what I want to cover in this video. We're have more of these, but I think that gives you an idea as to how you can approach the solving of a problem with and without having these APIs pre -implemented, pre -provisioned, or available for you to begin with. That's about it. Let me know, let us know if you have any questions.

And we'll talk soon. Thanks. Bye bye.