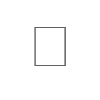
**Transcript**

September 25, 2025, 7:31PM

 **Justin Pope** started transcription

 **Justin Pope** 0:03  
Story. So um.  
So um.  
To be honest, it's not gonna be much of reading the ADA of story because there's not much to it, but.  
We will be quickly transferring over to Figma and going over some of the designs that I put together and make sure everyone's kosher with that. And then as we go through the design, feel free to ask questions and make sure everyone's up to.  
On par with what we need to do. This is going to be a lot technical in nature, but the end result is lifestyle snapshot is not gonna break and it's gonna support PDF uploads that we need in.

 **Charlie Bradley** 0:52  
Bye.

 **Justin Pope** 0:53  
The product performance and care story that will be.  
Next in line based off what we're doing this Sprint. So without further ado, we are refactoring the image entity that I got scope creeped into the lifestyle snapshot and we are going to make the appropriate changes to.  
For it to handle any and all content, um, for the time being, we are just going to, um, make the necessary changes to just accommodate PDFs.  
And go with that, with that synopsis of what this is, what's going on with this? Is there any questions?  
OK. Um, I'm sorry, Walter, you're good.

 **Walter Martinez** 1:51  
Perfectly.  
Yeah, I'm good.

 **Justin Pope** 1:56  
OK, uh, so.  
The description basically says that.  
I put a blurb, but just define the entity which the design document will or design Figma board will explain a lot more of what that definition is. Rob, I see notes of future uses.  
PDS for other documents, images used in things question mark. What else? Some of those questions are going to be noted in the Figma board and hopefully explains some future uses and we can talk through what other future uses.  
That we can go so.

 **Robert Hobbs** 2:44  
OK. Thanks.

 **Justin Pope** 2:47  
Outside of that, everyone ready to switch over?  
All right.

 **Charlie Bradley** 2:52  
Stewart.

 **Justin Pope** 2:54  
So here's the Figma board. Um, the.  
Approach I took with this design is first try to explain with as much details as needed what our current domain is based with images and then from that.  
What do we need to do to convert it over to be able to take on content and what we mean by content right now, so.  
Just trying to do a high level overview of image and not explain too much of the details. Feel free to read this on your own time, but simply.

 **Charlie Bradley** 3:39  
Sure.

 **Justin Pope** 3:47  
We have this image entity. We have image attributes. Currently we have validation on mime types for the image. We also have image attributes. Currently it's just image type. We're currently only using license snapshot as that.  
That's the only entry point. Some things of future use that was already baked into this is potentially are two easy cases that we just.  
Bearing that we have time on the road map, we can reconfigure organization images as well as theme images to also use this with simple modifications.  
Um, a note on we go that route. I think the we.  
Bearing that or when we embark on doing that, we would need to come back and define if there are more attributes to define those specific entities. So for instance organization images, right now we have logos in the.  
Missing product image. I would suggest that another image attribute type should embody those two different organization image types and consolidate that domain.  
Into the content attribute and as well As for theme. Right now there's only one theme image type for dashboard. I would also recommend that we probably need to think about how to embody.  
That image type within the image attribute.  
After saying all that, did I just go over everyone's head or does everyone understand the technicality of that?

 **Walter Martinez** 6:03  
Yeah, I'm I'm following.

 **Robert Hobbs** 6:04  
You're just talking about creating new new content type types that are domain specific.

 **Justin Pope** 6:05  
OK.  
Right. Yeah, for the.

 **Robert Hobbs** 6:16  
So something like like builder dashboard image could be a type I mean.

 **Justin Pope** 6:17  
Right.  
Right.

 **Robert Hobbs** 6:22  
Um, let me think for a minute here.  
So um.  
Those types can be useful. I don't know if those are if you're talking about making those part of the image attribute type structure. That's what you're suggesting as opposed to tagging them with something generic.

 **Justin Pope** 6:38  
Like.  
I would.  
Maybe I need to have a or we need to have a conversation of what you mean by tagging because I I would, I would think that this attribute entity does.

 **Robert Hobbs** 6:55  
Yeah.

 **Justin Pope** 7:03  
Exactly what you would say tagging does? Or how does this not do what you're thinking?

 **Robert Hobbs** 7:06  
Well, yeah, so it it may be fine. I'm what I'm trying to say is that if we have an image in this content system and one image is.

 **Justin Pope** 7:19  
Mhm.

 **Robert Hobbs** 7:22  
A lifestyle survey image choice or something like that. A lifestyle survey option, right? And another image is a buyer is a builder dashboard background image.

 **Justin Pope** 7:32  
Mhm.

 **Robert Hobbs** 7:39  
Right. It's important to know.

 **Justin Pope** 7:39  
Mhm.

 **Robert Hobbs** 7:44  
Each of those categorizations while still also knowing, recognizing that it is an image.  
So you need both because an image has attributes that are unique to the file type and the other one is more of a domain classification system. So if we've got that covered here, I think that's fine if we don't have the idea of covering both of those concepts for a piece of data.

 **Justin Pope** 7:54  
Mm.  
Right.

 **Robert Hobbs** 8:12  
yeah then it might need to be modified.

 **Justin Pope** 8:17  
So when you are talking about things that define the image itself, does.  
So looking at just I'm looking at how the image entity is constructing. Now we have image data, image, mime type, file name, content hash and length in bytes.  
Does is there any other uh?

 **Robert Hobbs** 8:43  
Yeah.  
Image mime type may cover it.

 **Justin Pope** 8:48  
OK.

 **Robert Hobbs** 8:49  
In other words, image my type may preserve the underlying file type, essentially information that we would need to do certain kinds of processing on it, and then the attribute would have the kind of the domain.

 **Justin Pope** 8:57  
Right.  
Right.

 **Robert Hobbs** 9:07  
Category. Is that kind of the idea?

 **Justin Pope** 9:09  
I.

 **Robert Hobbs** 9:10  
Or actually it does image attributes have other things like resolution or let's say like PDFs for example, we're always interested in like knowing the page counts. So would those be different attributes as well or?

 **Justin Pope** 9:15  
Oh, so.  
Sure, right.  
In my in my design I tried to make image attribute as flexible as possible. So like a potential I I feel like a potential future is that we utilize.

 **Robert Hobbs** 9:26  
Yeah.

 **Justin Pope** 9:41  
A attribute type specific to some sort of domain model and we could serialize a class that handles a lot of the properties to that specific domain into the value of this image attribute.

 **Robert Hobbs** 9:55  
Mm-hmm.

 **Justin Pope** 10:01  
So say, oh gosh, like.  
So you're talking about like resolution. We could throw in a new image attribute type to define a specific image.

 **Robert Hobbs** 10:08  
Yeah.

 **Justin Pope** 10:21  
Properties or categorizations and then have a class that defines those properties and then they are just serialized into the value property here.

 **Robert Hobbs** 10:24  
Mhm.  
That's fine. Yeah, that's cool. I just think I like we're usually interested in capturing like like you know in other. And again, we won't consolidate to this right away, but like we when we upload, when we upload PDFs to.

 **Justin Pope** 10:43  
Right.

 **Robert Hobbs** 10:49  
When when we upload PDFS to the GPC for example, right now we're capturing some metadata like page count, maybe file size becomes important as a as an attribute and maybe and then file name is usually an attribute. So all I'm really saying is like for each of these different kinds of content types that we're.

 **Justin Pope** 10:52  
Mhm.  
OK, yeah.  
Mhm.

 **Robert Hobbs** 11:08  
Going to support out of this store. Having the ability to model those kinds of attributes per MIME type, perhaps it seems like a good idea. That's all I'm really saying. Whether it's serialized or not is totally fine. Whether it's flattened out as attribute values doesn't really matter to me.

 **Justin Pope** 11:10  
Right.  
Sure.  
Right.

 **Robert Hobbs** 11:28  
I just want to make sure we're looking forward enough to have a spot for it, and it seems like we do, right? Seems like you kind of have that, so.

 **Justin Pope** 11:33  
Uh, so again.  
Yeah, I tried to model this. I did not directly model some of those attributes that you're talking about and we can come back whenever we are ready to tackle those kind of situations and refine the domain entity to accommodate those, but for right.

 **Robert Hobbs** 11:44  
Yeah, yeah.

 **Justin Pope** 11:58  
Right now either we could slap them into attribute or we can just extend the entity to have more properties for those particular attributes we would want to accommodate for. So I I think we have paths.

 **Robert Hobbs** 12:09  
OK, yeah, I don't know how many of those we want to capture on the PDF upload for specifically for like product performance and care. I I figured like for sure file name and stuff like that and probably like the fact that it's a PDF.

 **Justin Pope** 12:14  
For sure.  
Mhm.  
Right.

 **Robert Hobbs** 12:28  
And you know, OK, it might be an interesting exercise as part of the work if it doesn't add a ton to try to grab some other metadata and flange it into the the structure to kind of test it out. But if that blows it up too much, I wouldn't want to do it. But if you said like, well.

 **Justin Pope** 12:28  
Right.  
OK.

 **Robert Hobbs** 12:47  
Let's go ahead and parse the page count out of it and see if we can store that and then recognize, OK, there may be other things we want in the future, but this is kind of like a test of the attributes structuring. Yeah, cool.

 **Justin Pope** 12:50  
Mhm.  
Sure, sure. Um.  
As I continue working on this, if I see I have time, maybe I can try to do some of that. But for right now, I don't know if I'm gonna be able to get to that, but I'll try to keep that in the back of my mind.

 **Robert Hobbs** 13:15  
Mhm.  
Sure. And and some stuff too, by the way, I just want to make clear like some of them can be nice to haves for later stories like we're like the the PDF upload story is obviously blocked by the refactor that we're doing. So it's OK to say, well, I've got it, I've got a plan in mind, this is what it is, but I've staged it out so that that's.

 **Justin Pope** 13:24  
Right.  
Right, right.

 **Robert Hobbs** 13:36  
Gonna be a follow up text story. That's fine. Yeah, totally cool.

 **Justin Pope** 13:38  
Right, well I before I'm done with this refactor, there are certain points that I I would like to expand this design to accommodate for the future. For instance, I did a very basic pagination query for.

 **Robert Hobbs** 13:52  
Mhm.

 **Justin Pope** 13:58  
Images which we're not utilizing now, but we could. So just thinking about what changes need to be put in place so that the gallery could be used as patch foundation. Also the content management in the VDS app or VDS admin app.

 **Robert Hobbs** 14:11  
Yeah.

 **Justin Pope** 14:17  
How should we approach that is another thing we should think about because yeah.

 **Robert Hobbs** 14:23  
Yeah, I think we're going to need our own view for for managing the content library like yeah, and then other views kind of like what we have now for picking an image for the lifestyle snapshot would really be like a filtered query on it.

 **Justin Pope** 14:29  
For sure, I agree.

 **Robert Hobbs** 14:39  
It's like get me all of the things that are images that also have this domain attribute or whatever. So OK, cool. Thanks.

 **Justin Pope** 14:40  
Right, agreed.  
Yeah.  
OK, so let's move. Uh uh um.  
Anything else to note? So a lot of this, the way the domain works now is gonna be preserved in the.  
Refactor. A lot of it is just going to be name changes and then additional support for a type on the content entity. So images is going to evolve into content.  
Notice I did a lot of name changes. If you see in the image entity there was like image data, image bind type. A little to say the least, so we can just draw.

 **Robert Hobbs** 15:38  
Mhm.

 **Justin Pope** 15:46  
Off that and just have data.

 **Robert Hobbs** 15:47  
What? What was? What was? What was redundant? Sorry. Oh, content type and mime type. Or which one?

 **Justin Pope** 15:50  
So.  
Right. So the the like on the image entity you have an image mine type like we we don't need image on that property right?

 **Robert Hobbs** 16:09  
Um, because you you're gonna have a content type that describes it anyway. Is that the idea?

 **Justin Pope** 16:11  
Or am I?  
That and just like so if you're going to read it's going to be image dot image mine type like why be repetitive on the property.  
That that's all I'm trying to say.

 **Robert Hobbs** 16:27  
Mm-hmm. You still have mime type in this structure.

 **Justin Pope** 16:29  
So regardless.  
Right. Agreed.

 **Robert Hobbs** 16:33  
In in the content table and so that's gonna have values like.

 **Justin Pope** 16:36  
Uh huh.

 **Robert Hobbs** 16:40  
Like the ones I'm seeing on the right where you have content type and mime Tye validation.

 **Justin Pope** 16:46  
Right. So I guess let me explain. So type is now going to be added to content and that is where we can initially have image and document because that's what we want to support.

 **Robert Hobbs** 17:01  
I see.

 **Justin Pope** 17:03  
And based off that type, there are going to be specific mime types that correlate to those content types. So for image we're going to have JPEG, JPG.

 **Robert Hobbs** 17:03  
OK.

 **Justin Pope** 17:20  
And PNG to correlate to an image, while for a document right now we're gonna support PDFs.

 **Robert Hobbs** 17:27  
PDF. Yeah. And then maybe later Excel, Word, whatever. I get it. Um, OK.

 **Justin Pope** 17:29  
So.  
Right. And then we could even further out types to videos if we ever really want to dive into that. But like right now we're just do image and document.

 **Robert Hobbs** 17:37  
Right, Yeah.  
It's an interesting.  
Yeah, I could see maybe using down the road having even though it's kind of silly for a content table URL.

 **Justin Pope** 17:55  
A URL MM.

 **Robert Hobbs** 17:56  
A URL type, which is kind of silly, but we do we do specify URLs in places. So I don't know, maybe that's a stretch for this. We maybe it's unnecessary because it's not really like binary data, but but anyway, just a string.

 **Justin Pope** 18:00  
Theoretically, yeah.  
It's just a string, but I I I definitely see see your thought, but yeah.

 **Robert Hobbs** 18:12  
OK, so and these, yeah, and are these actual tables that I'm seeing or is this just the entity map?

 **Justin Pope** 18:18  
So what? So if you notice for content type, I have a content type enum. That enum is C# code that's going to be defined.

 **Robert Hobbs** 18:29  
Yeah.  
Yeah.

 **Justin Pope** 18:34  
So you have made comments of how you wanting these things exposed in our gosh in SQL in the database and I.

 **Robert Hobbs** 18:47  
In the database, yeah.

 **Justin Pope** 18:51  
I'm just gonna go with like I think this should just be code enforced because if a new content type comes in and we don't support that in the domain like it's not gonna.  
Mean much, right? So I'm just trying to define what kind of values do go into that string and how that's more or less being enforced by our codebase.  
And I have. I'm pretty sure I butchered all that, but does that make sense?

 **Robert Hobbs** 19:28  
No, I understand what you're saying. I just, I I think we will run into trouble down the road when third parties are asking us for um.  
Are asking us for the scope of data and I can never answer that without going to the code base. Then I have problems. You know what I'm saying? The other thing I'll just say too is that if you if you do make it database driven that it means that.

 **Justin Pope** 19:43  
Mhm.  
So I.

 **Robert Hobbs** 19:57  
Um, given that you're able to upload a file of any type, you don't have to code specifically for it or make a change.

 **Justin Pope** 20:05  
I'm sorry, say that one more time.

 **Robert Hobbs** 20:08  
Yeah, like if we decided to let people upload files from their machine of different types that are currently not supported, given that uploading a file of some type.

 **Justin Pope** 20:21  
Mhm.

 **Robert Hobbs** 20:25  
Is kind of a browser thing for the most part, right? Or it's part of it, or it's an upload control. I don't know why that couldn't be data-driven. Like starting tomorrow, we're going to support uploading Word files, right?

 **Justin Pope** 20:32  
Right.  
So.

 **Robert Hobbs** 20:41  
Like like I get that like parsing up the metadata is different for that versus a PDF. So I do get that side of it like you can't have it be completely data-driven, but I I think we can move on. Justin, I'm just pointing out that like if I start getting.

 **Justin Pope** 20:41  
Right.  
OK.

 **Robert Hobbs** 21:01  
Requests from third parties, which they're always mucking about in our database. Then my response either ends up having to be OK, let's make a table for it so that they can get to it or.  
Be more painful response. Let's wrap it in an API end point so that they can ask for the the content types, right? Those are kind of my only responses, you know what I'm you know what I'm getting at?

 **Justin Pope** 21:26  
I.  
Well, I was gonna suggest exposing this in an endpoint that they can read.

 **Robert Hobbs** 21:34  
OK, and and then that would be fine as well if I can say OK, no problem, open postman, call this endpoint. You can get the content types by this query.

 **Justin Pope** 21:37  
OK.  
Yeah, so yeah, my my initial thought is we can expose the content type and also the mime types and I feel like that could also enhance our front end development when so.

 **Robert Hobbs** 21:57  
Mhm.

 **Justin Pope** 22:03  
When we are in the gallery view of uploading an image to a lifestyle snapshot that is hard coded right then and there. So if we ever come back and say hey I want to expose or now I want.

 **Robert Hobbs** 22:04  
Yeah.

 **Justin Pope** 22:20  
What image type isn't here? I don't know SVGS, right? That's theoretically an image type. We would have to, right? We would have to edit this, go to the front end, edit the import logic to accept SVGS and then.

 **Robert Hobbs** 22:24  
Mhm.  
Yeah, and we actually have that in a different view.

 **Justin Pope** 22:40  
Done. If we just expose an endpoint to read this, then it's just one and theoretically done because now our front end is just saying hey, what's the valid mime types for an image? And it would say these are the valid mime types for an image and then we're done.

 **Robert Hobbs** 22:48  
Mhm.  
Mhm.

 **Justin Pope** 23:00  
So I regardless, those are again nebulous ideas that aren't concrete yet. So trying to move on with image attributes converting over to content attributes.  
Nothing is really changing in its schema. All that is really changing is additional data to support the document type. So notice now.  
Content attribute type enum. There's now a document type and then the document type is going to tie to content attribute type document type enum of product performance and care because those are the documents that we want to upload.

 **Robert Hobbs** 23:53  
Mhm.

 **Justin Pope** 23:55  
I have additional ideas of attributes that we can add to with the product performance and care story, but I will reserve those thoughts to that story when we get to it.  
So moving into domain note changes, really a lot of just renaming. One point I want to make is when creating the image repository.  
We are also doing a unique index on file name.  
Mulling it over, I don't know if having a unique index on file name is gonna do what we really want it to do, because right now we query if we're trying to upload a new image with the same file name.  
We just query the existing image now and then present that as the image you want to use. I don't content hash and file name do the same thing.

 **Robert Hobbs** 25:02  
Terms of the hash, is that what you mean?

 **Justin Pope** 25:11  
So not only is it unique content hash, it's also a unique file name. So my my thought is when we get to organization images, Taylor Morrison already has a logo.

 **Robert Hobbs** 25:14  
Mhm.

 **Charlie Bradley** 25:23  
So.

 **Justin Pope** 25:25  
In our repository, if someone else comes back and tries to upload another Taylor Morrison logo, like if we already have that file name, you're not updating any image, you're gonna just continue to use.

 **Robert Hobbs** 25:38  
Even if the image is different.

 **Justin Pope** 25:40  
Even if the image is different, yes. So what I believe we should do is just drop the index. We'll remove the functionality in the use case and just rely on the content hash.

 **Robert Hobbs** 25:42  
Yeah.  
Yeah.

 **Justin Pope** 25:56  
Um content hack.

 **Robert Hobbs** 25:57  
Although, yeah, yeah, sorry. No, no, go ahead.

 **Justin Pope** 26:02  
Uh, I I think everyone more or less knows. So um.  
I don't know explicitly what the content, what hashing does, but it's basically condensing data down into a defined string per se, but that's.  
Helps process the data instead of going over the byte array. In practice what it does, it just makes sure the exact image isn't being utilized in the image repository. So it's not it's not like.  
Gonna do if it's similar, it's like if it's the exact same image, so just wanted to be clear that's what that's doing.

 **Robert Hobbs** 26:48  
Is that hashing process the same for documents? Is there any reason to think it would be different? Yeah, OK.

 **Justin Pope** 26:52  
It's gonna go through the same. It shouldn't be different. We should still have a byte array and we're just throwing that through a.

 **Robert Hobbs** 26:57  
Mhm.  
So that would work on anything that is a byte array in terms of its data. OK, got it.

 **Justin Pope** 27:04  
Right. Well, if I understand it correctly, you can throw anything to be.  
Hashed really. So you could throw, you know, a class in there and as long as it's instantiated, you'll hash the data.

 **Robert Hobbs** 27:15  
Oh, all right, cool.

 **Charlie Bradley** 27:26  
Like.

 **Justin Pope** 27:26  
Because like hashing turns into like a security thing, so you can hash it, pass it along with a specific key and as long as you have the same key you can. I I don't know what the terminology, but like you can basically.  
Uh, unfold it right? And you would get the same data in and out, right?

 **Robert Hobbs** 27:45  
Mm.  
Got it. I have to run, unfortunately. Yeah. No, no, you're fine. I just wanted to say one other thing. I noticed you have some other changes needed over there and it feels like those expand the scope.

 **Justin Pope** 27:53  
That's a huge tangent and I'm sorry, OK, um.  
Yes.  
Yes.

 **Robert Hobbs** 28:08  
To other parts of the system beyond lifestyle, snapshot and um.  
And and this product performance and care PDF upload I assume.  
And so those would not be part of this refactor, right? Those would be future refactors.

 **Justin Pope** 28:26  
I I think.

 **Robert Hobbs** 28:31  
Or do I read that wrong?

 **Justin Pope** 28:32  
No, I think this needs to be done within this refactor. So you know the domain of this isn't just back end domain like it goes all the way into our front end, so for a.

 **Robert Hobbs** 28:42  
Mhm.

 **Justin Pope** 28:48  
If if you have the time like option one.

 **Robert Hobbs** 28:49  
It's it's fine I think.

 **Justin Pope** 28:53  
And and I think this is more of a question for specifically Daniel and of like what do we want to support long term or you know what is better for the system?

 **Robert Hobbs** 29:03  
Yeah, no, no, it's fine. I don't want to get. I think I got hung up on the idea of image repository as being something we already have in use out there generically to like fetch images. But in this case, you're really just talking about what we built for the lifestyle survey.

 **Justin Pope** 29:15  
No, yeah.  
Right, this is strictly just.

 **Robert Hobbs** 29:19  
I hope. OK. So yeah, my key thing is to just make sure it's contained like the regression testing we have to do is really just limited to lifestyle survey and then product performance and care and then we can like start folding other pieces of the app into this to to leverage it. Is that?

 **Justin Pope** 29:26  
Correct, yes.  
Correct. Yes, that is correct. The I'm not like other image stuff that existed pre lifestyle is not being changed, is not being modified. The image repository I noted here is strictly the repository that is fetching the image entity that repository will.

 **Robert Hobbs** 29:36  
Do I have that right? OK, cool. OK.  
Yeah, separate. OK.  
Sweet.  
Uh huh.

 **Justin Pope** 29:56  
Do follow suit would need to be renamed to be contents repository because that's what it's fetching and then it's basically at that point is we can utilize the use cases as is.

 **Robert Hobbs** 30:03  
Gotcha. Gotcha.

 **Justin Pope** 30:13  
Because we can just modify the use case to factor in just fetching and modifying content with the type of image type like that's one approach and then let that flow through our system as needed.

 **Charlie Bradley** 30:24  
OK.

 **Robert Hobbs** 30:28  
Mhm.

 **Justin Pope** 30:31  
Or it's now we need to do a full refactor to be fully encompassing of the entity of content type, and then everything needs to be refactored to adhere to the content type.

 **Robert Hobbs** 30:47  
Gotcha. It does. I don't think I have any input on that. So I think that's more of a team thing like you, like Daniel and Reed should certainly weigh in on it. I'm sorry I got to run, but I really appreciate you throwing this together and hopefully I apologize if I derailed you too much with questions. I was just trying to.

 **Justin Pope** 30:47  
Does that make sense?  
OK.  
Sure.  
No, I understand.  
Yes.  
No, we got tomorrow, so feel free to bombard with more questions.

 **Robert Hobbs** 31:07  
Get my head around some of it. So yeah, sure. OK. No, no, I I think I'm good and I appreciate it. Thank you. OK, bye.

 **Justin Pope** 31:15  
OK, sure thing. OK, bye. Daniel, do you have time to continue talking through this or do you need to drop off?

 **Daniel Arwe** 31:27  
Um, I have a little more time. I probably should get make some more progress on what I'm doing. But yeah, let's let's talk whatever we need to talk about.

 **Justin Pope** 31:35  
OK, so do you understand the predicament and why I laid out option one versus option 2?

 **Daniel Arwe** 31:44  
Not fully.

 **Justin Pope** 31:45  
OK, so.  
All right, so let's.  
Option one is to keep image use cases as is and reach into the content entity specifically only on the content type of image image.

 **Daniel Arwe** 32:00  
OK.

 **Justin Pope** 32:10  
Does that make sense?

 **Daniel Arwe** 32:11  
Image type image or a content type image.

 **Justin Pope** 32:13  
Yeah, comage type image the if if we are going to go down that path that would us and then mean we need to create use cases to be specific to the document content type.

 **Daniel Arwe** 32:16  
OK.  
OK, so we would need endpoints to fetch images, which we already have, and we would need new new endpoints to fetch documents.

 **Justin Pope** 32:40  
Right.  
Right.

 **Daniel Arwe** 32:43  
Rather than going through a single set of endpoints for a some piece of data, whether it's an image or a PDF document.

 **Justin Pope** 32:49  
Either.  
Right. So the the question is, if we go to a single entity of content, does the whole application need to look at that?

 **Daniel Arwe** 32:53  
OK.

 **Justin Pope** 33:05  
On its content or basically can we diverge the be? I guess I don't know how the design but like we have use cases of fetching images and then we're just.

 **Daniel Arwe** 33:20  
Mhm.

 **Justin Pope** 33:21  
More or less duplicating of use cases, fetching and modifying documents. So if we.

 **Daniel Arwe** 33:29  
Mm.

 **Justin Pope** 33:33  
If we go with like the only like negative outcome is just like double the code because anything you're using to fetch images, you need to duplicate to fetch documents and more or less keep that kosher or whatever our.  
Need is for fetching documents versus images for this refactor. It just kind of means once we got this entity renamed and done.

 **Daniel Arwe** 34:08  
Mhm.

 **Justin Pope** 34:09  
Then we just modify the use cases to adhere to the content type of image and we're done like we don't need to do that.  
I.  
I don't know if that's the best approach because we are basically now refining the entity to a specific.  
Content type, which I guess at that case, like I wish Eric was here to see if he had an opinion on that, whether or not we need to know, don't diverge, keep it all within the content domain.  
And just be specific on your fetching and content management. Um.  
Because if we say, oh no, we're gonna do, we're gonna keep that. That means OK, we need to rename all those use cases to be content specific. We just still we can just utilize all those use cases for any kind of content we upload, but then the.  
The the work that needs to be done is on the querying and like preparation of that data. You have to be like OK, this is a document.  
It needs to have this kind of document attributes, all that yada, yada, yada. But I don't know, I digress, so.  
Do you have an opinion on one path forward versus the other? But and like if you need to maybe step away and think about it a little longer, that is fine as well.

 **Daniel Arwe** 36:05  
So are the endpoints and the use cases we created for the lifestyle snapshot? Can they be?  
Are they agnostic? Do they care whether they're returning an image or a PDF or some other form of document? Some other kind of document, like a Word document Monday? Maybe? I don't know. Like how much specificity do we need there?  
Versus like, is it just like, hey, send me back a byte array of some sort and a type or like, I don't. I don't really know how that all works under the hood, so that's where I'm a little bit confused.

 **Justin Pope** 36:40  
Right. Well, so okay, so let's look at it. So I think my

 **Daniel Arwe** 36:41  
Mhm.

 **Justin Pope** 37:00  
So.  
Yes, this. So this right here is how.  
For a specific image, this is what actually returns the document or the the image I to my understanding since we are passing back data and the mime type for that data.  
It it could be any supported mime type, right? So if that data is a PDF and the mime type is correct for that byte byte array, this endpoint the way it's constructed can return an image or a document. It doesn't matter.

 **Daniel Arwe** 37:45  
OK.  
Mhm.

 **Justin Pope** 37:52  
Right. So um.

 **Daniel Arwe** 37:53  
Mhm, mhm.

 **Justin Pope** 38:00  
I would, I would, since this is what if we choose to have it be agnostic of the specific content types.

 **Daniel Arwe** 38:14  
Right.

 **Justin Pope** 38:16  
I I would recommend we create a new content controller with these endpoints, but they're gonna change in signature, right? So that's the reason why there's gonna have to be front end changes.

 **Daniel Arwe** 38:23  
Mm.  
Yeah.  
There will be front end. OK, so front end changes for lifestyle snapshot, but only in the sense that we're changing what we're pointing at in the front end.

 **Justin Pope** 38:40  
Uh, there. Well, who? Um.  
We would have to modify, so another nuance, there's the get image URL helper.  
I would propose that we also so that that's what preparing the actual URL to get the the image on a DTOI would propose we would need since we are going away from a specific.

 **Daniel Arwe** 39:13  
Mhm.

 **Justin Pope** 39:19  
Entity content type Um.  
That should follow suit and be like content URL helper and then point to whatever the get content URL is.  
Does that make sense or I'm just like spinning in circles?

 **Daniel Arwe** 39:40  
Yeah, yeah, I understand. So it sounds like a lot of tedious changes, but not a lot, a lot of functional changes. It's really just like dotting all the I's and crossing all the T's to make sure something doesn't break is what I'm hearing.

 **Justin Pope** 39:49  
Right, I I'm like.  
Right. That is entirely what this is. There is like.

 **Daniel Arwe** 39:56  
OK.  
OK.

 **Justin Pope** 40:01  
It is very little functional change to say the least, I think because because I'm sorry.

 **Daniel Arwe** 40:05  
OK, so from a test.

 **Justin Pope** 40:28  
Correct.

 **Daniel Arwe** 40:28  
OK, so then all we're talking.

 **Justin Pope** 40:29  
Walter, feel free to try and hit the endpoint, you know, naked if you want. But I I think that's all that's been defined as what would mark this as a success of just it doesn't break our sticking hands.  
Um.

 **Daniel Arwe** 40:49  
OK, so it sounds like that's the agnostic path is probably the way to go, and it sounds like the other path is just a lot of code duplication, which obviously is not great.

 **Justin Pope** 41:00  
Right. I think that's the the main. So yeah, the main issue with retaining is like the way is then we have to do a lot of code duplication in the future.  
If we do a content agnostic, that just means we do, yeah, a lot of tedious changes. So I would propose, I'm sorry, go.

 **Daniel Arwe** 41:26  
OK.  
Well, I was gonna say the good news is yes, we have to go back and make tedious changes to existing functionality, but at least it's only the lifestyle snapshot and not a bunch of other stuff that's pointing at this, because that would be exponentially more changes that would have to be made and and retested.

 **Justin Pope** 41:37  
Correct, correct.

 **Daniel Arwe** 41:43  
Um.

 **Justin Pope** 41:44  
Right.

 **Daniel Arwe** 41:46  
So yeah, it seems like it's worth it to bite the bullet now before this. We have too many things pointing at this content store and.  
Just take care of it now and keep it agnostic so we're not duplicating a bunch of code unless you can get really creative with like base classes and like reusing some stuff. But sounds like it would be easier to keep it agnostic as long as keeping it agnostic doesn't cause us a problem. I mean if it the endpoint doesn't care if it's returning an image or a PDF document or a Word document.  
Document then I don't see the problem as long as we and I but I do agree things need to be renamed to reflect like hey, we're not just returning images, we're returning potentially other types of files. So yeah, I think we just have to be super like thorough and careful and I think testing will also reveal if we've made a mistake somewhere and.

 **Justin Pope** 42:25  
Correct.  
Yes.

 **Daniel Arwe** 42:36  
Or overlook something, but I don't want to treat Walter's time flippantly, so that's, you know, I was a concern, but it sounds like we have test cases already written that just need to be rerun, correct?

 **Justin Pope** 42:38  
I believe so, yes.  
Mhm.

 **Walter Martinez** 42:50  
Yeah, that that's right.

 **Daniel Arwe** 42:51  
OK.

 **Walter Martinez** 42:53  
Also, with all the test cases that are already written, maybe what I can do is to find any other kind of ways to.

 **Daniel Arwe** 42:54  
All right.

 **Walter Martinez** 43:03  
I mean try to adapt to new files. I mean try to different files since in lifestyle is not sure I guess the ones that we have it, they are limited. So maybe we can like discover more options that I can try just to make sure we're.  
We're covering most of the scenarios or yeah, making sure that it's not breaking.

 **Daniel Arwe** 43:21  
Mhm.

 **Walter Martinez** 43:26  
The the the actual feature, let's say.

 **Daniel Arwe** 43:28  
Yeah, so you're talking about like trying to push a PDF up to lifestyle snapshot or something and see if it'll see if it allows that to happen. And I feel like the file keeper's gonna restrict he he's saying just push up like a file type that's not supported and see if it.

 **Walter Martinez** 43:34  
Yeah, that's right.  
Mhm.

 **Justin Pope** 43:39  
I'm sorry, say that again.

 **Daniel Arwe** 43:46  
Breaks or see if it accepts it. And now I'm. I'm thinking the file picker should limit what we're able to see at least. Now you can always change the file picker to like, hey, show me everything, but yeah.

 **Justin Pope** 43:56  
OK, I'm glad you bring that up. Do you? I I think. I think with that file picker we should.  
When we open up the gallery, we need to expose endpoints to.  
Query what valid mime types are and then our file picker is a bit more solid in design and then it's.

 **Daniel Arwe** 44:29  
Right. OK, So what I'm.

 **Justin Pope** 44:34  
It's it's never gonna be a question of oh, are these mime types correct? It's no, it's we're querying the domain explicitly for the correct mime type.

 **Daniel Arwe** 44:45  
Yeah, so the front end just just asks the back end like what are my acceptable mime types for this specific content type? Like hey I'm I'm doing a I'm doing a lifestyle snapshot image. What are my what are my acceptable types? And it would say JPEG, PNG and that's it.

 **Justin Pope** 44:52  
Correct.  
Right.  
Correct, correct, yes.

 **Daniel Arwe** 45:03  
And then if we ask it for product performance and care, I would say, well the only acceptable format is PDF and and then we would limit on the front end what's what's able to happen there.

 **Justin Pope** 45:16  
Yes, agreed. Um.

 **Daniel Arwe** 45:16  
Correct. OK, cool. Sounds like we have a plan.

 **Justin Pope** 45:22  
OK.  
Any other what? What else you got, Walter?

 **Walter Martinez** 45:34  
Well, not right now. I mean I I don't think we, I mean I think everything that it's already discussed, it's it's always covering what I also had in mind. So I don't think that we have any edge or we we're edge case right now. What I will be doing it to like I will be re re reading everything and making sure that everything that it's discussed.  
Already there and we didn't miss anything. So if that's the case I I will be writing test cases for that if if it's needed, even though what I what I will be doing is reusing the ones that we just have. And yeah, if if it's anything weird that I can spot, I will definitely reach out to you guys to to.  
To make sure that we we got everything covered. Yeah, I don't. I don't. I don't think we're missing anything so far. I think we're good.

 **Justin Pope** 46:17  
OK, great. Yeah, this has been recorded. I will post the transcript into the war room. I think I might give ChatGPT a once over and summarize a lot of these points.

 **Walter Martinez** 46:24  
Mhm.

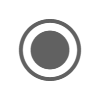
 **Justin Pope** 46:34  
And then we'll go from there. Thank you all for this time. It's and yeah, thank you all.

 **Daniel Arwe** 46:38  
All right.  
All right, thank you.

 **Justin Pope** 46:44  
All right. See ya. Bye.

 **Walter Martinez** 46:46  
Thanks. Take care.

 **Daniel Arwe** 46:46  
OK. Bye.

 **Justin Pope** stopped transcription