EPISODE 1230

[INTRODUCTION]

[00:00:00] JM: Product teams sometimes double as data teams. They struggle through import errors, scrub long and complicated data sheets for consistency, and map spreadsheet fields on step three of a long instruction document. Data structuring and synchronization is a very real problem that product teams regularly have to overcome. Flatfile uses AI-assisted data onboarding to eliminate the repetitive work and make b2b data transactions fast, intuitive and error-free. By drawing on the experiences of their thousands of users, Flatfile automatically learns how imported data should be structured and cleaned. Their product saves product teams a ton of time setting up data and syncing it across workspaces so people can focus on their data instead of fixing it.

David Boskovic is the co-founder and CEO of Flatfile. He was previously the platform architect at Envoy and CTO and co-founder of the startup Rainmaker. David joins the show today to talk about Flatfile and his vision for the product in the company.

[INTERVIEW]

[00:00:57] JM: Guys, welcome to the show.

[00:00:59] DB: Thank you.

[00:01:00] EC: Thanks for having us, Jeff.

[00:01:03] JM: You guys work on Flatfile, which is solving a very specific seemingly niche problem that is actually experienced by a lot of people, which is data onboarding. Can you explain what data onboarding is?

[00:01:18] DB: Yeah. B2b SaaS products or largely b2b software products all have this problem of being empty boxes that require customers to bring data to them in order for that box to do something powerful. And what we saw when we started this company was that almost all these data exchanges are happening entirely manually. Either customers are being given spreadsheet templates to fill out and get into the exact right format and then provide back to a very simple sort of upload UI or it's going all the way to implementation services where serve implementation teams are helping customers get their data into these products.

Either way, it's an incredibly labor intensive product. It's an incredibly labor intensive problem. And the existing tools, so if you look at the entire data stack, your ETL tools, they're designed for data experts, data scientists, anybody working on the data stack. When you look at this problem, it's primarily a customer-facing problem with a non-technical or certainly not data scientist in the process. So solving this problem with the product we've built, the challenge was to really create a data tool that could be used by people who have no idea what a data problem is, right? They have no idea what a boolean or a string is. These are foreign terms to them. How do you build an ETL product that these people use?

[00:02:48] JM: So who is the target user? Like who is the data user at the company that if it's not an engineer, who is the person that is doing this kind of labor? Can you take me inside the life of a prototypical user?

[00:03:02] DB: Yeah. So we have a product we call Portal, which is embedded inside of a SaaS application. Think of it, if you're familiar with sort of embedded offering, think of it like a plaid for data or a Stripe Checkout for data. You have a SaaS product and you have an import button in that product. Let's take the example of a CRM and you want your customers to be able to upload their contacts into that CRM. Once they click that import button, Flatfile brings up an overlay on top of that application that allows the person to upload a file, map data in, label it, correct it and get it into the product quickly. So the user of Flatfile is actually the customer once it's embedded in the solution. The implementations team or your services team there is there to facilitate the customer sort of getting their data into the product, but largely we want the customer to be able to do it themselves. So it's sort of in addition to not building for a technical

audience, we're building for a highly unpredictable audience. It could be everything from one of the most interesting use cases of Flatfile is with a grocery store who is using it to get aggregate data from local farmers who are providing a list of the vegetables and things that are available on a given week and loading that in. All the way up to a Fortune 500 company that's trying to onboard data from their insurance brokers.

Sort of the user persona is highly variant and the skill levels of these people are highly variant. As we've focused on building out this product more and more we found that we have to sort of optimize the lowest common denominator, which is somebody with almost no technical skills. We expect basically just the most minimal sort of ability to understand a spreadsheet type UI and be able to interact with that.

[00:04:55] JM: So there are products that are embedding this kind of import feature into them, like you mentioned a CRM. Can you explain the integration story a little bit more like and maybe you can give an example of a typical kind of user or a typical customer?

[00:05:14] EC: Yeah. So if you think about a typical customer of Flatfile, they're the recipient of the data who we sell to because they're the most incentivized to get that data in their system. And the way the process works is usually they've already designed some sort of system or solution. So let me take an abstracted example. We've got a customer that works with teachers and school systems on planning out student rosters, recording grades, sort of getting all this highly disparate student data into their system. They've already got that system designed. So they've got their data model set up. They understand how data is meant to be used in their system. That's not how data comes in to the system. And that's really what Flatfile does, is we serve as a bridge between that data coming in and the ideal state of that data as it's usable in that software system.

So the way that Flatfile actually gets deployed from a practical perspective is our customers who are the data recipients just tell Flatfile, "Hey, this is how the data needs to be structured when it comes into our system," and then our solution is what guides this we'll say non-data expert, but a subject matter expert through the process of getting their data into that solution.

So in the case I mentioned with the company that works with school systems, oftentimes it's going to be a teacher or a school administrator who is gathering these, again, disparate sort of documents that contain rosters of students or courses and then dropping it into our customer's sort of education management system via Flatfile.

[00:06:54] JM: Okay. So let's say I'm uploading a CSV to some website that has integrated with Flatfile and that CSV has some errors in it. Like I've got email addresses in the wrong place or I've got ZIP codes in the wrong place. How does Flatfile figure out that there are problems with that CSV?

[00:07:17] EC: Yeah. So that's actually one of the benefits of using something like a Flatfile is that we go beyond just like what the machine's requirements are of the data, but rather like allowing our customers to share in human readable language, "Hey, this is what's going on." So as David was mentioning before, it's a really sort of simple intuitive interface that has human readable messages that guide people towards the right solution. So if you were a teacher and you uploaded some data on students and let's just say the system's expecting sort of a student registration number that's not included. We would highlight that and say, "Hey, it looks like this student registration number isn't included. You need to include this." And that need for inclusion is informed by how our customers sort of design Flatfile to fit within their data model.

Again, our customers input the data to Flatfile is that data model. What is the business logic and rules around data that you accept? And then their customers input the Flatfile is the data itself. And then their understanding of, "Oh, okay. I need to get the student registration number." They're the subject matter expert. They know what the student's name is, the registration number, where they live, their address, etc. They have all that information. So they're in the best position to make a decision about how and what to adjust and edit if there are any issues with requirements pulling that data into the system.

[00:08:47] JM: Tell me about some of the engineering challenges in building this kind of import tool.

[00:08:54] DB: One of the biggest challenges is building a complex ETL product. And for the audience that doesn't understand the ETL terminology, it's extract, transform, load. It's getting data out of any source format, getting it into the right shape and loading it into a destination format. Traditionally this is something done by data scientists or data engineers. If you try to condense that problem down, it's the same problem being solved for an audience that doesn't have any of those skills. So how do you take these deeply technical challenging problems and put them in front of this different audience? And I think that's for us the core technical problem as well as the core user experience problem. How do you ask a simple question like what encoding is this to somebody who doesn't know what encoding is? Period.

For us, we might sort of step back and take a first principles approach to solving that technically as well as solving that in the UX. For example, we might put in front of the user three different versions of the file and ask them which one looks right. Now we've solved for this encoding problem in a way that you don't have to know the underlying technical problem to actually answer it correctly. If I look at three versions of this file, I see one of them is corrupted and one of them isn't, I'll click on the one that is – And move forwards. And we can sort of filter through a lot of questions similarly, right?

Another example of this is Booleans, getting somebody to map data into a boolean format for the destination system when they don't even understand what that term means. So we'll ask them, instead of asking them, "Is this true or false?" We'll give them a visualization like, "Does yes mean checked and no mean unchecked?" And we'll ask that for maybe any language in the world. Where we might be able to predict that yes is checked and no it's not, we might not be able to immediately predict that for like other languages that we haven't yet trained off of.

And this sort of comes to the last point, which is automating as much as possible. If we can predict that yes means check then no means false, we can move on and not turn that into a complex data transformation thing. But then we get to reserve the user's time for things that are more substantial. So we focus on learning every time somebody answers one of these questions and trying to put that into a training model and then make recommendations to

users. It's so much easier for a user to accept a recommendation than it is for them to go and engineer something in the first place or think about the solution in the first place.

And what we find is that there's a vast majority of these data problems are low-hanging fruit. We get to actually solve them quite effectively sometimes up to 95% of the time effectively in an automated fashion. Reducing the amount of time that users have in front of the spreadsheet from hours to sometimes less than a minute just spraying that data into the product.

Be that underlying technical challenge there is quite substantial, which is you might have a thousand different UI experiences for the user that based off the different data problems that the user has to go through. How do we put something in front of users so they only get the three that are most relevant to them but also have the ability to confirm the end state to ensure that it's accurate?

Human loop engineering has now started becoming a lot more common as machine learning companies have expanded, but one of the few times I've ever seen sort of the human in the loop being somebody who has no context of the technical problems. Traditional human loop engineering would be finding an expert who's going to label an image and flag it as a traffic cone. We're going to put this image in front of somebody. We're going to loop them in. We're going to ask them this question. We're doing the same thing for the end user and we're trying to frame the question in a way that they can answer, train us on it and us move forward without them ever really knowing that they were a human in the loop of a machine learning process.

[00:13:03] JM: What are the other common data problems of companies that are accepting these kinds of imports?

[00:13:12] EC: I think one of the most challenging problems is often just like the structure and the labeling of data. If you think about it, there're so many systems out there. You've probably interacted with one before where they said, "Hey, here's a template. You've got to go take your data and shape it into this template or this format." And really like we believe that our software

should be smart enough to do that for us and especially if it's seen other examples of real humans like you doing that previously.

And so a good example or a practical example of this would just be like a foreign language. So let's just say you're a business in Mexico and you sign up for a software solution that was built in the United States. You might have a bunch of customer data or data in general that's living in spreadsheets and all of the column headers are in Spanish. Name is nombre. Well, it seems kind of counterintuitive to make you as a human tell the machine over and over again every single time you go through this process that, “Yes. Hey, name does mean nombre.”

And so really those data problems, like David mentioned, can be kind of low-hanging fruit for a system that just gets to leverage a little bit of human intuition to help, for example, appropriately structure the data. Other challenges include things like custom validation logic. So I’m guessing you've gone through the experience before of like using one of these software solutions that allow you to pull data into it and then 30 minutes later receiving an email message saying, “Hey, here's the 137 rows that had issues. And then maybe if you're lucky some like technical printout of all the data issues that exist.”

Well, instead of doing all that validation on that data after the facts that is unique to that particular business or use case, you can pull that up front and say, “Hey, when you check this box. When you say, “Hey, this is –” Let's just use the insurance example. This is a smoker, right? Like if you check that box for that particular subscriber to the insurance saying that, “Hey, if you have this person's a smoker, we're going to make this other field required. You're going to have to let us know the length or duration of how long they have been using tobacco.”

And so those are examples of things that are like data problems that very quickly turn into sort of like organizational workflow problems and ultimately cost centers for the business when, really, when you're getting data from a customer you should be able to leverage that data and turn that into a profit center by using that data in your system.

[00:15:55] JM: So tell me more about what you have to build to accommodate the wide variety of data infrastructure that your customers might be operating over.

[00:16:08] DB: When you think about sort of the core problem we're solving, almost always the destination of this data is a product, oftentimes a REST API or a product database. So it's pretty rare that it's a sort of data warehouse or a traditional sort of data pipeline infrastructure. And so oftentimes what we're trying to do is get the data into the exact shape that the API needs it to be in order to process that ` – Or for it to be processed into the product. So there's sort of a twofold problem. One is we need the user to clean up the data and get the data valid and then we need the data to get into the exact JSON shape or GraphQL shape necessary to load it into the product. And then we need to do that at scale.

So if I'm loading 10 records, we can just use your API. If you're loading a million records, we're probably going to break everything. And so there's a combination of both managing the ETL side of it as well as the sort of queuing infrastructure and loading infrastructure necessary to get this data into a product even at scale. Largely we don't get to benefit from a sort of downstream data pipeline that would normally exist sort of in a traditional ETL world. So we've had to sort of build and engineer these things into Flatfile sort of as core functionality, which actually allows us to now adapt to almost any sort of internal data system. Like maybe I want to get the clean data out of Flatfile actually just as another CSV file in the right structure. Maybe I need it in XML, because Flatfile is sort of taken on and owned some of that traditional data queuing and streaming infrastructure. Now we can actually serve that to a customer wherever they are. If that does happen to be a database or a data pipeline, we're handing that data through in a clean way. And if you don't have that, we already have it sort of set up.

And I think that one of the underrated things and sort of what we're doing here and the problem that we're solving is the fact that building this yourself is a lot of work, right? Your MVP solution is if you're just going to take the most minimal path, you're just going to say, "Hey, here's a specific CSV file template. We're only going to accept this template. The data has to be in exactly the right format, otherwise we'll reject it." That's where almost all SaaS products start. And then they start expanding on that, "Well, we don't need the dates in the exact right

format. Let's allow for some of this to be like – Maybe we can use three different formats on the dates." But largely the prep has to happen in advance.

And so like each one of these steps, whether you take away like data transformation step, or you take away the data queueing step, we have large like late stage SaaS companies that haven't figured out a good sort of data streaming system yet, like they limit their customers to an import of a thousand records because their system breaks otherwise. And it's incredibly interesting just how complex each of these sort of components is and now we've had to sort of solve for each one of them. But in doing that, have taken an incredibly big problem entirely away from a SaaS product and put it into a simple sort of three line widget you can connect to your existing API and move on providing your customers a really great experience.

So I think like super interesting as an engineer and a product designer, so you just keep looking at how much of this entire workflow we have to build in and how much of this expertise we have to develop internally. To develop this deep sort of understanding of performant queues, we have to develop this understanding of user experience around technical data issues and we have to be able to provide a usable SDK for almost all types of applications. So it's been incredibly fun and interesting like working with each of our customers to unpack these technical problems and be able to meet them wherever they are.

[00:20:06] JM: What are the problems that you found not feasible to solve in terms of data onboarding?

[00:20:14] DB: Yeah, there is a long tail of problems. Right now we aren't doing things that require OCR. So for example, PDF with a bunch of invoices in it, or a bunch of images that need to be detected as to sort of what's in the images. That's not sort of our area of expertise right now. We're focusing almost entirely on what we consider to be sort of Flatfile data, hence the name of the company. So whether it's a CSV file or an Excel spreadsheet, the data is structured reasonably enough and we can take it from there and start normalizing that into a common format.

Over time we do expect to enable the exchange of almost any data that can be turned into a structured format. So we're starting to work with partners who, for example, might be able to take a folder full of 7,000 payroll stubs and turn that into structured data where Flatfile would actually pick up and then allow the continued process onboarding. Right now that's where we sort of draw the line though, and it'll be interesting when we do tackle that.

[00:21:21] JM: Now there's a wide variety of different front ends that people might have that they're building around this data onboarding tool. So do you have to build components for Angular and React and, I don't know, PHP templating? Like how do you handle the wide variety of frontends?

[00:21:43] DB: Yeah, we do basically. We have a common JavaScript library that can work with any frontend, but we also have a native React component, a native Angular component as well as expanding into a lot of these other frameworks. The closer we can get to giving you a sort of one line to drop in your application, the easier it is to implement Flatfile. On the backend, we provide a lot of support around how you can ingest that data. You can actually just take the data client side and post it to an API that's already authenticated or you can pull from our API, you can get it pushed by a streaming web hook sort of opt into whatever the shape of your application is and sort of get your ideal approach. You can opt into – Getting the data that way.

Overall we've gotten to a place where Flatfile can be implemented in a couple hours. We have seen customers sign up at 3 pm and have it in production by the end of the day. Now that's an incredibly skilled customer. For larger customers oftentimes it'll be something that goes through a Sprint, but at the very most, this turns something that is an extensive sort of build out into something that you get to benefit from the 300 plus customers that have used Flatfile but in the product that you drop into your solution.

The incredibly interesting thing for us there is that we get to learn across customers, and this is one of the most exciting things to me that got me to quit my job and actually fundraise for this company, was realizing that out of our 21st 20 pilot customers, we had 20,000 end users that had gone through uploaded data, labeled it, corrected it. We got to watch everything they were

doing. And one of the most substantial problems in building any sort of machine learning company is access to subject matter experts to train off of, and we got that incredible opportunity to observe how people make these decisions. And so yeah, we can avoid having to build out long tail algorithms as we build out this solution and we can just actually get to a place where we suggest the right solution or close to the right solution. And because the subject matter expert, the person providing the data, is able to confirm whether or not our suggestion is right, we get to facilitate that in the UI embedded in your native application easily. And so part of the React component or Angular component is the button, but then the rest of that is the actual entire workflow that goes into asking users these questions that allow them to train Flatfile and train your system to be able to automate most of the processes customer after customer.

[00:24:22] JM: I'd love to get a better picture for the machine learning side of things. So it's sort of hard to imagine, like you have this import button that lots of different websites use for importing different types of data and the schema is always different for different applications. So how are you training a system to do machine learning across like a totally heterogeneous system like that?

[00:24:49] DB: Yeah. So when you think about like the long tail problem here, let's pick an example just like first name. How many different ways can we see a first name labeled and structured in an incoming data? So at this point we've seen some 25,000 variations on the full name, first name, etc. There're two things that we want to figure out. One is can we detect from whatever labels are available if there is a header, right? F name, first name, whatever it's going to be, that this is a likely first name instead of a last name? The second thing is can we detect based on the pattern of the data that's going through that whether or not it's actually a first name? Often times this is the core problem with data coming in, the full name will actually be in an either unnamed or poorly labeled column.

So Flatfile can – But with the volume of data that we're seeing, for any sort of type of data, we can observe the patterns of data and train off of that. So we can build out models that allow us to detect what the data looks like in addition to how it's labeled in the source system. So even

without a header or even without sort of any labeling that's coming with the data, we can detect that the pattern of data looks similar to data that has been assigned to first name or full name before. So that way we can tell that F name actually is first name, not full name. F name obviously is an unhelpful label.

You might look at that example and go wow that would be reasonably easy to solve with an algorithm, but the reality is there're tens of thousands of these sort of ontological definitions of data and Flatfile doesn't want to go and build algorithms for each of those. So can we detect based off some sort of historical analysis of the volume of data and the pattern of data going through what the most likely correlation of this is?

The next layer for us is to invest in the actual transformations. So if we watch a user sort of fix a few cells, right? Let's stick with this full name example, right? I'm bringing full name. I want a first name and last name out. Now we could build an algorithm that splits those things apart or we can watch the user correct a dozen of those and manually copy the last name to the last name column and then give them a learned prediction on what they might want to do with the rest of the data. That's a better solution for us because that allows us to serve across all of the verticals that we provide this offering to. Allows us to serve them even if we haven't yet built something out for that type of data, whether it's an ISBN number for a publisher or a classification for an e-commerce company or a finance system or payroll. Once we've seen a few thousand users go through and upload data, it's built enough of a history of understanding that pattern of data that we can actually train off of it and make reasonably good predictions to the end user as to what they might want to do there.

[00:27:48] JM: Can you tell me how engineering is arranged across the company right now? You're pretty focused on this solution of customer data onboarding. So I'm just wondering how the different team members are allocated across that application.

[00:28:04] DB: One thing we did early on is we made a determination to have a homogenous stack across both frontend and backend. So we're a fully TypeScript shop. Our backend is in TypeScript. Our React application is in TypeScript. That allows our engineers to be really

versatile whether they're building out user experience or whether they're designing the next sort of highly performant queue on the backend. And in addition to that we have hired our entire engineering team as full stack sort of engineering experts at a senior level. That allows us to actually have a very flat sort of design on our engineering team and allow people to work on projects that are most valuable at any point to the business as well as sort of code review and each other's work effectively.

Going forwards, we're going to start forming out some product teams, but that's sort of one of the core benefits of the way we've structured the talent on the team is that it allows people to choose sort of what products they want to focus on. It lets people sort of focus on the thing that they're most excited about. They're all capable of doing anything. So what is the thing that really gets you excited and gets you energetic around this problem? And everybody has a different thing there. So as we scale we're going to keep leaning into sort of the advantages we've built on the team with these incredible engineers who can sort of do anything and give them a lot of latitude to sort of self-define their area of focus and do the thing that is highest leverage for them in the business.

That sort of is also just an artifact of our culture. We really want to see individual stakeholders in the business making substantial decisions about either architecture design or product experience. And so we try to keep all of our engineers and our product designers very close to each other and collaborating around the problems that are being solved. So that, A, we have this high-level of empathy for the customer, but everybody's sort of on the same page and able to think about the problem solving together. Yeah, broadly a pretty flat architecture in our engineering team and we'll try to keep it that way.

[00:30:13] JM: Where do you want to expand the product line to over time?

[00:30:19] EC: One of the things that David was kind of alluding to is this idea of actually like reducing the number of product interactions that happen. So if you think about it, a lot of product teams are oriented around getting people to use their product actively as much as possible. The way we design our product's objectives it's all related to the value that the

customer of ours receives. So is the customer and their customer receiving the appropriate value of Flatfile? And if we extrapolate that out, really at the end of the day the idea when I mentioned the Flatfile is no interaction. Flatfile kind of takes care of everything behind the scenes for you. And so we kind of maintain that as a high-level vision for the product is, “Hey, could we actually serve this as just pure value?” There might be some initial configuration that our customers do. There might be some ways for us to understand our customer’s data model and their business logic. But at the end of the day, the less you can do in Flatfile, because Flatfile is just doing it for you, the better. And that's what we're doing by sort of garnering all of this information about how these different subject matter experts address these data problems is really to kind of feed into that longer term vision. And so really from that perspective we don't necessarily add more products, rather apply our product in a lot more different places.

So just one practical example of this would be using Flatfile is just an SDK that's part of an existing ETL process. Let's just say you're a global real estate company. Wouldn't be great to have the intuition of tens of thousands of real estate agents who have onboarded real estate type data across all of Flatfile’s products and systems integrated into your data pipeline that's pulling from, let's just say, a set of market listings? So you could automatically enrich and normalize that data as it's flowing into your system. So I wouldn't expect necessarily to have a different technical product so much as different packages or solutions based on how we've been able to learn about different segments of data and be able to learn about how to apply that human intuition in those various circumstances.

Again, that can look like something like an SDK. Maybe a solution or a package of Flatfile that's focused more on internal data exchanges as opposed to data exchanges on the edge. But right now our focus is sort of rounding out this solution for data onboarding, and we've got this initial product that we offer that's portable that's an embeddable data onboarding solution. Now we've started working on the solution called Concierge, which is more of a collaborative data onboarding platform. So when you're running data onboarding as a project, a good example of this would be like from my previous experience working at Epic. We would deploy Epic’s electronic medical record system at a new hospital system, but oftentimes what that meant was it's a pretty extensive project where we were involving multiple clinics and hospitals and

making sure to aggregate and normalize their data and running a cut over into that Epic system.

So when you manage data onboarding like a project, and also there might be some sensitivity around the data, and when you actually run that cut over, that's what the Concierge solution is helping to solve. So not only are we addressing those data issues that exist, but also addressing some of those workflow concerns that come up when you run into this type of product.

[00:34:01] DB: To add to that, one of the really interesting challenges for us is that the sort of core success metric of our product is the least number of interactions possible. So the ultimate success of our product is actually not to be present at all. To illustrate this, we're working on getting to a one-click import. I drop a file in. It recognizes maps and corrects all the problems in that file and gets that data into the product. Now, obviously, there always be times where there're material issues and you have to loop the user into that flow, but it does allow us to drive towards this sort of visionary goal of doing away with our own product as the end goal of building the product.

[00:34:46] JM: Do you have any perspective on the changing nature of the overall data stack? There are a lot of new companies in the world of data engineering and I just love to get your respective perspectives on how the world of data engineering is changing.

[00:35:03] DB: Yeah. I can weigh in on this. I know Eric has thoughts as well. The interesting thing for us is we actually play very much laterally to almost everything in the data engineering space. Your traditional products, whether they'd be your products like a Trifacta or anything that's managing the ETL process or getting – And there's new players like Fivetran, DBT, obviously just raising incredible round. These are all serving the data team inside the house. And I think broadly what we're seeing is a lot more of that, right? A lot more competition in this sort of red ocean of data products surveying the data teams. It just indicates that there's a lot more to do there.

Flatfile is sort of in a little bit of a different position where we're handling data exchange at the edge of the business, and that area is entirely blue ocean right now. Flatfile is leading the way in providing a product that allows you to actually exchange sort of a relatively automated fashion data with a vendor or a customer rather than opening up Excel or starting a big services project. So I think what we'll see is more expansion like Flatfile on the data space. I think there's the incredible amount of competition as well as innovation sort of inside the castle if you would. But I think we'll start seeing a lot more innovation outside of the castle like Flatfile on this sort of edge-based data exchange or high-variance data exchange.

[00:36:38] EC: Yeah. And I guess I'll add to that. I'm probably going to contribute to the overuse of this quote, but the classic software is eating the world by Marc Andreessen. It's absolutely true. Software's taking over every sort of industry and segment that exists in the market, but software really feeds on industries. What it breathes is data. And you need to have data in those software systems and have it be usable in order to get the most out of them. And so what you're seeing as a result is this like rapid democratization of access to data tools and systems and analytics software, everything from opinionated BI solutions to sort of like self-serve analytics tools. And really for us one of the biggest gaps we saw here was like, "Hey, there's also like data in a workflow." Especially when two companies are interacting, that's not really being well-served. Like we're still sort of manually addressing this, and folks might be using a non-purpose fit software to help them through that process, but it's still at the end of the day a very manual lift.

And so just to lean in what David said and be more explicit, like a lot of these data problems are moving away from the centralization in the CIO's office, but rather across the entire organization. And for us what that means is focusing on sort of like the CTO and COO's office where you have these product and customer teams that are dealing with these data challenges day-in and day-out and they don't want to wait on the CIO's office to build some overarching system to solve their data needs at scale. And that might not even be possible given the variance of the types of customers and data that they work with.

[00:38:23] JM: Have you seen any applications of Flatfile that have surprised you?

[00:38:30] EC: Oh man! I love to tell you about my favorites. I don't know if surprising is the right term, but maybe validating. One of my favorites is a grocery store. So we've got a small grocery store chain that's using Flatfile right now in a system that they've developed to keep track of all of their different advertisers in store. So basically they want to send out notifications to all the folks who are working on the grocery store floor to change out things like end caps and coupon codes. But what they need is they need to understand like, "Okay. What are all the different contracts that we have?" And since all those contracts are coming from different data systems, normalizing that data can be particularly challenging. And what Flatfile lets them do is say, "Hey, you're a store manager and you run a contract with like a local sort of retailer or food provider distributor. Go ahead and load it up into the system via Flatfile, and then that way we can notify your team about exactly when and what needs to be placed either out on the shelf or in some sort of advertising."

[00:39:36] DB: And another really interesting one is this jet fuel management software. So every time an airplane has to fuel up on the tarmac they have to type in the amount of fuel that was loaded and the type and all this goes into these really legacy pieces of equipment. And at the end of the day these tarmac operators produce a report that includes all of the planes that fueled and has to get loaded into this general system. So they use Flatfile to import data from those systems into that sort of aggregation.

We see a lot of this across over 100 different verticals, whether it'd be aggregating data from mining machinery into a central system. Or if you're a fortune 100, aggregating data from a ton of contractors or working with your supply chain. I think there's a – One that was absolutely exceptional was pest. Eric, was it a pest management software company?

[00:40:42] EC: Yeah. There're all these different unique applications. And when I say it's not surprising, but rather validating, is there's this internal mantra we have, which is every company in the world. Now when you think about it, more and more of these phase businesses are exchanging data as part of their relationship together, and that's not surprising to us. That's a core thesis of our business. But what's really validating and exciting to us is seeing the

realization of these unique cases coming up. So whether it's a pets management control company, a fortune 500, wine distributor or something in between. A good example, one of our customers recently, I can share this publicly because we've done a case study together, [inaudible 00:41:22] is helping with COVID testing, tracking and distribution. And one of the challenges for them is that they've got all these folks who are medical professionals or subject matter experts on issuing tests in the fields, but not necessarily great on gathering data and normalizing it to get it into its system to know whether or not they need to order more tests or to understand those test results correctly. And so what we're doing is really empowering a lot of these exchanges between businesses to let them get back to doing what they should be doing. Like we want the COVID testers spending their time giving COVID tests, not necessarily wrangling data into a templated format to get it in the system to issue a report.

[00:42:05] JM: Cool. Well, as we draw to a close, do you guys have any additional thoughts on the sector that you're working in like customer data onboarding, no code applications or just general thoughts on the future of software?

[00:42:23] DB: I think one of the things that we're seeing is that I love this quote from David Sacks a few years ago, and it's that, “A good heuristic for building SaaS companies is looking at any problem that you're taking care of in a spreadsheet and building a product for that.” I think we'll continue seeing a lot of that. No code is tackling either sort of the spreadsheet problem or solving these componentized problems inside this explosion of SaaS products.

As we start seeing more SaaS products emerge, you're going after more and more sort of focused and verticalized enterprise spaces. And so this means that sort of technology that previously had its last innovation 20 years ago now is getting a new layer of innovation. We're starting to see this in legal tech and across an incredible variety of verticals where there's now sort of both the attention from venture capital as well as the interest and ability to innovate in these really interesting areas.

So I think like what I love about Flatfile and I love about building on Flatfile is we get to be part of all of these stories. We get to be part of everything from company solving for COVID

research or environmental research, all the way through to huge enterprise challenges. And it's changing at its core the economics of using data. And when you change the economics of anything, something in the world is affected, right? If you make it cheaper to access research on the environment, what's the downstream impact of people using that research to positively affect changes in the environment? There's also a downside, right? You may make it easier to access data. How can that be abused?

And so I think as we scale up this company and continue to make it easier and easier for our customers to use the data that they have, how can we be thinking about sort of both the positive and negative effects of changing the economics of data? So I think we're incredibly excited to be building this business and we take that responsibility pretty seriously. And one of the things that I love to share here is we're trying to hire up dozens of engineers right now. We're looking for people who are really interested and applying their skills towards something that materially sort of changes and impacts the way business operates. So incredibly excited about that and, yeah, I think that's my synopsis.

[00:45:00] JM: Great. Well, thanks to both you for coming on the show. It's been real pleasure talking.

[00:45:05] DB: Thank you. Thank you for having us, Jeff.

[00:45:06] EC: Thank you.

[END]