EPISODE 1013

[INTRODUCTION]

[0:00:00.3] JM: Every company has a large volume of routine data workflows. These data workflows involves spreadsheets, CSV files and tedious manual work to be done by a knowledge worker. For example, data might need to be taken from Salesforce and filtered for new customers and piped into MailChimp, or perhaps you need to sort all your customers to find only the ones who have spent more than $50.

These data workflows might require some basic knowledge of SQL or an understanding of how to make an API request. Not everyone in the world knows how to execute these somewhat technical commands. A software company can be slowed down due to a shortage of technical analysts, who have the necessary programming skills to build these kinds of data workflows.

Parabola is a low code tool for building data workflows. Parabola lets the user drag and drop different components together to build an application without using a programming language. Parabola lowers the technical barrier for knowledge workers who want to build these kinds of data workflows.

Alex Yaseen is the CEO of Parabola and he joins the show to talk about the ideas behind Parabola and his goals with the company. It's a great conversation about a particular kind of low code tool; one that might also be quite useful for engineers.

[SPONSOR MESSAGE]

[0:01:32.9] JM: I remember the days when I went to an office. Every day, so much of my time was spent in commute. Once I was at the office, I had to spend time going to meeting rooms and walking to lunch and there were so many ways in which office work takes away your ability to be productive. That's why remote work is awesome. Remote work is more productive, allows you to work anywhere, allows you to be with your cats. I'm looking at my cats right now.

There's a reason why people still work full-time in offices. Remote work can be isolating. That's why remote workers join an organization like X-Team. X-Team is a community for developers. When you join X-Team, you join a community that will support you while allowing you to remain independent. X-Team will help you find work that you love for some of the top companies in the world. X-Team is trusted by companies like Twitter, Coinbase and Riot Games.

Go to x-team.com/sedaily to find out about X-Team and apply to join the company. If you use that link, X-Team will know that you came from listening to Software Engineering Daily. That would mean that you listened to a podcast about software engineering in your spare time, which is a great sign. Or maybe you're in an office listening to Software Engineering Daily. If that's the case, maybe you should check out x-team.com/sedaily and apply to work remotely for X-Team.

At X-Team you can work from anywhere and experience a futuristic culture. Actually, I don't even know if I should be saying you work for X-Team. It might be more like you work with X-Team, because you become part of the community, rather than working for X-Team. And you work for different companies; you work for Twitter, or Coinbase or some other top company that has an interesting engineering stack, except that you work remotely.

X-Team is a great option for someone who wants to work anywhere with top companies, maintaining your independence, not tying yourself to a extremely long work engagement, which is the norm with these in-person companies. You can check it out by going to x-team.com/sedaily.

Thanks to X-Team for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:04:09.6] JM: Alex Yaseen, welcome to Software Engineering Daily.

[0:04:11.3] AY: Thank you.

[0:04:13.1] JM: At every large company, there's a set of knowledge workers that work mostly in spreadsheets. Why has the spreadsheet been such an irreplaceable tool for so long?

[0:04:22.6] AY: Well, I think it's really the only tool for the past probably four years that's actually been built for those type of users. The knowledge worker that you're talking about, somebody's working in marketing, or sales, or operations, might be this really interesting analytical thinker with a deep strategic mindset. A lot of what would otherwise be a quantitative horsepower, but somewhere along the way didn't learn how to code, didn't slot themselves into the engineering path and they ended up in marketing or operations or sales and don't have a lot of ways to express that quantitative horsepower, or ways to work with data, unless things or tools are given to them.

Starting with early spreadsheets, I think one of the reasons they – the first killer feature for computers, it gave all of these people the ability for the first time to actually do things in the digital world.

[0:05:06.5] JM: What are the ways in which spreadsheet workflows are insufficient?

[0:05:10.1] AY: Well, the early spreadsheets were made to sum columns of data for financial statements and things like that. Today, people are running whole businesses on top of spreadsheets, where they're doing inventory management and marketing attribution, all kinds of detailed accounting workflows, things that a spreadsheet was really never built to do.

Some of the ideas of every cell of a spreadsheet being able to reference every other cell and having all these deeply nested formulas without much of the structure that a software engineer would apply to code, really makes them on these tangled mess that is the least good solution to a lot of these problems, but it's also about the only solution to a lot of these problems at the moment. I think that's why business users don't get really well served, but still use spreadsheets throughout every company that exists.

[0:05:54.9] JM: At some point, a given spreadsheet workflow becomes so complex that you would want an engineer to be brought in to create some internal application to replace the spreadsheet, but there's never enough engineering resources. Most of the time, the company just keeps the painful spreadsheet process. Is that accurate, or do you think that spreadsheet – do spreadsheet workflows get replaced by code?

[0:06:20.6] AY: Oh, no. I think that's totally accurate. Right out of undergrad, I worked in strategy consulting from – with a variety of companies from startups, up to Fortune 50 companies. Probably the biggest commonality that I saw was the amount of core-core business processes that your average exec maybe doesn't even know exists, but that are really holding up the whole company, are just running on top of spreadsheets.

One company I'll leave unnamed, Fortune 50 company, we were trying to dig into where the data that was feeding an entire call center worth of operations was coming from. Somebody told us it was coming from Craigslist. We said, “Oh, it couldn't possibly be on a Craigslist, the website.” They said, “Oh, no. This guy named Craig who used to work at our company five years ago made a spreadsheet and we call it Craigslist as a joke.” Nobody knew how it worked, but they were inputting data and it was feeding the operations of a 50-person a call center, just some random spreadsheet a person had made. I saw these things really throughout companies of all sizes.

I think you're right. I think part of the reason that is true is there aren't enough engineers in the world to work on all of the data and build all these processes that you need to be built. I like to say that even at Google, or whoever you want to think of currently as the best recruiter of engineers in Silicon Valley can't hire enough software engineers to do all of the data work they want to do, let alone all of the companies that can't even dream of hiring a Silicon Valley type of software engineer. I think that's part of the problem.

I think the other part of the problem is this Craig who made Craigslist, actually was a really deep subject matter expert at how to make call center operations work really efficiently for this company, and had spent probably a career developing that expertise. Some software engineer that they would bring in one day to replace I think, probably doesn't have that expertise. I think there's something really interesting about the person with the expertise being able to also be the person who builds the solution. I think in today's world that's very rarely the case, at least for full software products.

[0:08:15.5] JM: Right, because the subject matter expert is going to have some pieces of domain expertise that they don't even know how to express, but it's going to come out in whatever Excel model that they build.

[0:08:28.8] AY: Right, exactly. Hopefully, if they build it and build it well. That's only for the subset of people who know how to build Excel models. I think there's a even larger group of people who the word vlookup, or something that is common Excel task gives the nightmares and is a really scary concept.

Yeah, I think the person who becomes a subject matter expert in a lot of these areas is the absolute best product marketer, or best inventory manager, or person managing the call center operations doesn't really have the time to take on the overhead of sometimes it's even learning Excel, but certainly and not learning how to code and stand taking 20% of their time to stay fresh with the latest JavaScript frameworks and all that kind of stuff. They're really embedded in their deep subject matter expertise.

As a result, we end up with this world where either a lot of these problems go unsolved and the people who really have identified the most interesting problems to solve and the best opportunities to apply automation and the promise of the digital revolution and computers in general to their tasks aren't really able to do so. In the rare instances where they are, they have to write up some crazy requirements doc and get the buy-in of some engineers that their company hopefully can hire and retain and build software that will then have to go through some change management process and be maintained. It becomes this whole nightmare when all this person really wanted to do is just get their ideas that they had in their head out into the world and making the world a better place.

[0:09:48.1] JM: When you take the category of knowledge worker, there is this spectrum of technicality. On one end, you have the programmer. The programmer can create whatever they need to create with code. On the other end of the spectrum, you have the spreadsheet worker. Spreadsheet workers are still very intellectually capable, but they're nowhere near the freedom and flexibility of the programmer. What is between these two types of users on the spectrum? What is the semi-technical user?

[0:10:24.2] AY: We think about this a lot actually, because for Parabola, the company I'm currently founding, this user you're describing is our initial, earliest adopter and has been our earliest adopter. We call them a productivity optimizer, where they are just starting to dip their toe in the water of these more technical concepts. It's frequently somebody who once took a SQL course online and didn't really go much further, but knew that they thought that that was a thing they needed to do.

Someone who if you walked into a marketing team meeting at any sufficiently big company, had everybody close their eyes and point at the data person in the room, they'd all point at the same person. It's not on their job description, but they just become the go-to person for the slightly more technical, slightly more sophisticated things. They're really yearning for this opportunity to cross the gap from a non-engineer to an engineer and just haven't yet done so. Maybe to back up for a second, just that gap between non-engineer and an engineer.

I think you would lay out all of the types of users you were just describing, or people you're just describing as a productivity leveraged spectrum. Somebody who like the most tenured experienced software engineer in a few hours of really focused time can create this huge amount of economic output. They can do really amazing things, because they can build a thing on their computer and get it out into the world. The amount of value they're creating is the amount of value that people are getting by using the thing they created.

On the extreme other side, you have somebody who is doing a very manual labor type of role, where the amount of value other people get is the amount of time that they put into what they're doing. Most jobs in the world today are much closer to that manual labor side than they are to the software engineer side. This magical moment happens when you learn how to code today, where you jump from being the person who the amount of output you're able to create as how much input you put in to being able to create things to have a life of their own. We're in the business of helping people to jump across that gap. Ideally, we get everybody in the world down to that and a farthest other side of the spectrum to be able to jump across that gap.

[0:12:27.3] JM: You work on Parabola, which is a visual programming tool for working data sets and APIs and workflow automation. Describe the problem in a little more detail that Parabola is trying to solve.

[0:12:42.2] AY: What we're really doing at Parabola is we're giving people a platform to take their ideas, like what they want to do and turning it into something a computer to understand and execute on a set of instructions. That sounds like code. We have described that as a visual programming tool before. Frequently, we get taught, there's no code, a group of tools gets talked about a lot. I think you've had a few people on your podcast talking about them.

I really don't like to group us in, or talk too much about the no code world specifically, because I think that's a lot of talking about what it isn't, right? It's the core innovation there being that it's not coding. Most of our users aren't debating whether they should learn to code or use a tool. That ship has sailed for them. They're not planning on taking on all this overhead of learning how to code. They're looking for a way to take their creativity and their curiosity and their analytic capability and get it out into the world and show that they are the person who can create things and not just someone who has to be beholden to someone who doesn't know how to create that stuff.

We're giving these people the capability to be self-sufficient and be that subject matter expert we were just talking about, but actually also be able to then create the tool that solves their pain point, or the pain points they identify.

[0:13:56.2] JM: In Parabola, the user creates a flow. Can you give an example of a typical flow that a user would create?

[0:14:05.5] AY: Sure. These flows range in a wide variety of complexity. The simplest flow might be pulling in a data source, so maybe it's literally a CSV file for my computer, or maybe something I have in Dropbox, applying some amount of transformations to it. The simplest one would be I'm just a filtering out some columns. A much more complex one would be I want to run a sentiment analysis on each row of data and get some positive or negative sentiment score and all kinds of types of transforms in between those levels of complexity.

Then at the end, I probably want to send that someplace, or take some action on that data. I want to put it back into Dropbox. I want to send my boss an e-mail with the consolidated report I just created. I want to send a text message alert every morning with the ARR of my company.

Whatever type of action you want to take. For your more technical listeners, that sounds like an ETL process or something like that, where I think a lot of tools probably follow somewhere at flow, but what we really focus on is the middle part with all those transformations and the logic you can apply in the middle.

Any software program you're building has a view layer and a logic layer and a data layer. We really play a lot in that logic layer, but specifically on the rich logic you can build, not just the data connections on the other side. For in the Parabola world, those are more just the implementation details of of course, you have to get data in and out.

[0:15:22.8] JM: Can you give a more specific example? If I look at Parabola, you've got all these different connectors, like I can connect to Salesforce data, to a relational database and then create a CSV that dumps into Dropbox. I can take a Stripe data and do some e-mail marketing automation with HubSpot. I can make these kinds of workflows. Is there a concise prototypical example? Let's say there's a problem within an organization that somebody would implement typically in Parabola.

[0:15:58.7] AY: For sure. The biggest problem that we see that's really this painful just to see, let alone for the actual person is a really manual repetitive task that somebody is doing. Within a company, pretty much anybody in the operation side, or the marketing side, finance team, there's a set of tasks that are being done because of as we were just talking about, not having access to software engineers, not knowing how to deploy a better system, that these people are just holding everything together with the glue of them doing things manually.

Somebody gives them a CSV file, they do a bunch of clean-up on it, they're removing columns, finding or placing whatever that action is. They themselves every single day as head of operations of an e-commerce company, or a marketing analyst doing marketing attribution are spending hours of their life per day, or per week just manually cranking the machine when that's really not the promise of working with the computer. Whenever we identify really common areas there, those are the best use cases for Parabola.

To make them more concrete, we see some of the best use cases coming, particularly from e-commerce and retail, maybe manufacturing companies where they're working with large

amounts of data, tends to be items in inventory, SKUs, things like that, they are selling those things with credit card transactions. They have to mark it usually on the Internet to get people to buy their services and they then have to do some accounting with creating general ledger entries of all these transactions.

If you break those pieces down, those are operations managers doing a lot of inventory management workflows, where they have third-party logistics companies and various vendors and all kinds of different, pretty antiquated companies usually, that either don't have APIs, or they're not the APIs that a traditional software engineer would normally encounter. They're these really old, clunky interfaces that the data all needs to be combined together, some logic needs to be applied. If I'm that operations manager doing inventory management, I need to understand if I sold three items in my Shopify store, I need to decrement my inventory by three. But if I sold one special combo four-pack that I'm also selling, now I need to decrement by four every time I sell one of those things.

Just all these weird edge case logic that just gets solved by having a bunch of operations people just apply their time to solving these problems currently. People are able to automate those tasks with Parabola. Then there's really subject matter expert operations managers in that case could start focusing on how do we solve the next layer of complexity in our company and work a little bit more strategically, rather than spending all of our time manually operating this and holding things together ourselves.

[SPONSOR MESSAGE]

[0:18:47.7] JM: DigitalOcean makes infrastructure simple. I continue to use DigitalOcean, because of the low friction and attention to user experience. DigitalOcean has kept the experience simple and I can spin up a server in less than a minute and get high-quality performance for a low price.

For an application that needs to scale, DigitalOcean has CPU-optimized droplets, memory-optimized droplets, managed databases, managed Kubernetes and many more products. DigitalOcean has the flexibility to choose the right instance for the right workload and you can mix and match different configurations of CPU and RAM.

If you get stuck, DigitalOcean has thousands of high-quality tutorials, responsive Q&A forums and a customer team who treats customers respectfully. DigitalOcean lets developers focus on what they are building.

Visit do.co/sedaily and receive $100 in credit over 60 days. That $100 can be put towards hosting, or infrastructure and that includes managed databases, a managed Kubernetes service and more. If you want to get started with Kubernetes, DigitalOcean is a great place to go. You can use your $100 to start building your distributed system and you can get that $100 in credit for free at do.co/sedaily.

Thank you to DigitalOcean for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:20:22.5] JM: If you think about somebody operating a Shopify store, you mentioned that example of somebody buys three of something, isn't that information just taking care of on the back-end by Shopify? Why would I have to build some custom workflow that would need to be kicked off in response to an order for three pairs of socks?

[0:20:45.7] AY: Yeah. The amount of these workflows that are just slightly unique per company is incredible. That Shopify example is just one small type of thing. Imagine I'm working across multiple systems, each with its own interesting quirk. It's almost inconceivable that all of those systems will build every possible customization and workflow and checkbox and whatever to support every possible company that wants to work a little bit differently, or have some interesting competitive advantage.

The example of the bundled item you're just talking about as a marketing team for a butting direct-to-consumer e-commerce company is trying to innovate and do interesting new marketing techniques to stand out. They're not going to want to wait till Shopify releases whatever new feature to support their specific idea they just had. They just had a really interesting idea and they want to implement it.

If you were a software engineer, you would just build that thing yourself right on the spot. It would be some interesting growth test. You'd A/B test it, you'd see how it works and then you'd go from there. As that marketing person, you really don't have the abilities to do that. Instead, you do it really manually. You download a bunch of CSV files and you or an analyst on your team, or somebody crunches all that data and sends it off to be uploaded back into your inventory management system on the other side and that's the way you deployed you're A/B test.

I think interesting that that is surprising, because to a non-technical user, one of those roles at one of those companies, that's just day-to-day life. To get their interesting things done, they just have to do it the manual way, because they don't really have another way.

[0:22:11.3] JM: Right. If I have a Shopify site and let's say, every day I want to run a system that will calculate the item that has been ordered the most. Let's say it's a pair of socks, for example. I want to generate an e-mail marketing campaign, or a Facebook ads campaign for that item like, send an e-mail to all of the customer e-mail addresses I have that says, "These socks are hot. 10% off of these socks." If I have a programmer on my team that can build that, they can wire together some API requests. They can wire together a Python script and some Node.js stuff and they can create that automated e-mail marketing campaign.

More often than not, I'm not going to have a programmer available. I have somebody who's an operations expert. If they want to use Parabola, they can look at Parabola and Parabola has a interface for dragging and dropping different integration points, so you could conceivably take the Shopify endpoint as a drag and drop item, you could create maybe a MailChimp drag and drop item on this interface and you could draw an arrow between them, connecting them and put any logic between those arrow, on that arrow that you would want to and you could create that via a drag-and-drop interface, instead of a API code-based interface, which would open it up to visual programmers essentially.

[0:23:54.8] AY: Absolutely. If you really start getting into this, what does it mean to program, or what does it mean to code? That really is an API you just made internally for your company. It just depends on the abstraction level that you're thinking about. With parabola, we have all of those building blocks, whether it's connecting directly to an API and making an API request if

you want to configure that, whether that's working with one of our integrations we already provide to MailChimp as Shopify as we're describing. Then all of those transforms stringing this together, ends up looking like a lambda function.

If you're an engineer, you've now created a core piece of functionality that's repeatable. If somebody else in your team wants to open up that flow you just described, it looks like a flowchart. It's this tree of steps that are being taken, so it's self-documenting. Somebody else can say, "Oh, I see how this is working. We're first importing from these three sources and then we're joining all that data together. We're looking up our list of customer e-mails and then we're choosing to send e-mails to the people who bought this item within the past 60 days, but not within the past 30 days." You can just see that all very visually.

Not only can that person who is not technical create it, but someone whether technical or not, can understand immediately how it works, audit that it's being done correctly and go in and make tweaks when we want to three weeks later change that test just a little bit.

[0:25:06.7] JM: The example that I described, is that a realistic example? Is that something somebody could build in Parabola?

[0:25:11.9] AY: Oh, yeah. We have people using almost literally that exact flow. We call our templates and pre-built examples recipes, because a recipe for cooking something is something that most people are familiar with. They understand that it's a suggestion of how you could go about doing something. If you tweak it a little bit, maybe you could improve on it, maybe there's something unique that you want to do differently, more salt, less salt, whatever.

Anyway, so we have these set of recipes that we publish and increase and their users are publishing. Those types of examples are some of the most popular recipes that people will create as I'm doing an e-commerce store and I do want to run with those interesting marketing tests and send an e-mail, either about a top-selling item, or an item that's we know that we just ordered, but we haven't yet put up on our site and we want to start pushing it because we're about to have extra inventory.

We have some users who are doing some really interesting predictive reordering of inventory, because it's a pretty big problem for e-commerce companies running on pretty thin margins is you don't want to end up with just a huge amount of extra inventory for items that aren't selling.

If you have a huge amount of SKUs, that's very time consuming to understand how much inventory you have for each item and how quickly it's selling out. You might have enough interesting quirks in your audience and your product and your overall operations that has to be a very highly customized process, that someone who's familiar with the company is implementing. People build interesting flows around all types of that inventory management, probably is our most popular subset of use cases.

[0:26:36.1] JM: The e-commerce –

[0:26:37.2] AY: The e-commerce and the inventory management that you were describing, yeah. That was extremely spot-on, what you were just describing.

[0:26:42.9] JM: The integration points, so if in that example, if I want to connect to Shopify on one end and connect to MailChimp on the other end, how well developed are the APIs for Shopify and MailChimp and how seamless is it for you to provide a connector interface? What does the end user have to do to plug into those connector interfaces? How painful is it?

[0:27:06.9] AY: Yeah. For our most popular integrations, keeping in mind that we are targeting an increasingly less technical audience. We have first class integrations that are just a simple off flow. I click a little button that says ‘authorize’, I get my normal OAuth window that has my Shopify username and password I have to type in, or my MailChimp username and password I have to type in, just like a user is familiar with seeing with any tool they use. Then I'm immediately pulling my data into Parabola.

We understand the object types in a more user-friendly way than just a normal API endpoint. Rather than just my orders endpoint and my transactions endpoint, or whatever they have, or grouping those together into something that's a little bit more familiar for a user who is used to looking at the dashboard of one of those tools. We make that flow really easy.

For an integration that's maybe less popular that we don't yet support, we have a generic API connector that basically just sends HTTP requests. It's pretty robust as well. You can again, as a not too technical user configure connecting to an API, handling the pagination requests, which we do pretty intelligently, doing various kinds of authentication, whether that's very token authentication or OAuth authentication, we can connect to a lot of those types of things and enable someone who maybe is a little bit willing to read an API doc, but certainly not a developer to call an API, use an API and get access to what I think frequently gets called the API economy, without having to write a line of code.

Those types of use cases get really interesting as well. When we see a lot of people connecting to the same URL API, of course we'll want to make that a better experience for our user, so they don't have to configure this from scratch every single time. We've seen, I think we're into the multiple thousands of services that we've seen people connect with that API connector, which is pretty cool and I think people get quite excited about, "Hey, I've heard about these API things. I know they have a lot of interesting data. I know my company maybe put a bunch of time into creating a public API, but as a non-engineer I've never been able to really experience what that means," because not only is it difficult to send an HTTP request and make an API call.

Even if you just don't know how to figure how to do that, like with curl or something, how do you then work with the resulting data? Parabola not only makes it really easy to connect those APIs, but we pull it in, convert it into a nice tabular rows and columns of data like you would see in Excel and make it easy for somebody to start working with an API for the first time.

[0:29:28.8] JM: It's very hard to introduce new paradigms, like what you're trying to do. I mean, there's a number of companies that are attempting to reimagine how workflow automation works, or well, I guess introduced workflow automation in the first place, because I think this really is the replacement for a spreadsheet-based manual task. I mean, there have been attempts at this thing over the years. My understanding is that nothing has really stuck as much as spreadsheets and manual processes.

You see newer things, whether it's Zapier or Monday.com, or Airtable, Retool, all these different things are trying to change the paradigm. I see it as incredibly hard to change the inertia, because there's so much inertia around, "Look, I just need to get this done today. I've got a

million things to do. I would rather just stick to my spreadsheet workflow. I'm not going to adopt your new tool. It's impossible for me to have time to do that." How do you convince people to try the different paradigm that you're introducing?

[0:30:38.6] AY: Yeah, I think that's absolutely the case. I think that's probably the main reason why Excel, Google Sheets, essentially the same paradigm has been the most popular way for all non-engineers to do work for the past 30 or 40 years. I think there's a few ways of solving the problem. I think first part is there are those group of people that we were calling productivity optimizers, who are predisposed to want to up-level themselves. They're seeing people who are software engineers, maybe they're friends with them, maybe see them at their companies who are really operating at this level that is hard to keep up with and they want to be able to do the same thing. They want the same access to build their own ideas, not be able to hold into this tiny group of people in the world who know how to code.

I think broader than that, I have a technical background, but I also have worked in the finance world. I like to apply market force dynamics to a lot of problems. That's a lens I look at a lot of problems through. Market forces are really hard things to compete with. As this new group of whether you call them no code tools, or new productivity tools that you were talking about, have started to get some more traction and the companies that use them achieve – and the people at those companies who use them achieve substantially better results at the end. They're more productive. They're automating those really manual tasks and moving off to focusing on the much more interesting, more strategic higher leverage problems.

The person or the company who 10 years from now has refused to adopt that new way of working in a more leveraged way can't possibly keep up with a company that spent 10 years investing in how does every person, or a company start having this more strategic, higher leverage job.

I think to some extent, giving people access to these things and letting the market pull in that direction is something where I'm starting to see and I think whether it's a Webflow and a Bubble, or Airtable, or some of the other more project management tools, you have Notion and a Monday.com, they are – all of these companies together I think are really benefiting each other by if you try Airtable, or you try Notion, or you try Parabola and you realize, "Oh, wow. There's

this new way of doing work. I don't have to show up for work every morning and just spend six hours doing my really manual tasks. Instead, I can build a system that is imbued with that knowledge and logic and then move on to solving the next most important problem."

Once you've experienced that for the first time in any of the set of tools, you're immediately going to try to go explore some – the remainder of the set of new productivity tools. I think every new user that we show Parabola to, or that Airtable, onboards into Airtable becomes a user of this whole category. I think we're all helping each other and helping the world understand how they can benefit from working better.

[0:33:28.5] JM: Living in San Francisco, it's easy to convince ourselves that this is happening. I want it to happen. I think it should happen. I want the market forces to play out the way that you have portrayed them. For me, it's always tough to get perspective for what is actually happening in the broader world. Are people actually adopting these things to the extent that you need them to build a company around?

[0:34:03.9] AY: Yeah. Maybe two ways of looking at that. I think the more quantitative way is the vast majority of our customers and our users are not in San Francisco. They're in companies that have been manufacturing items that you maybe use on a daily basis, but never really think about what goes into that process and how do you procure all of the different materials and manage the time schedules of the people who are making those things and deal with the inventory management once you've made all these items.

Anyway, there's a huge amount of really smart, difficult, strategic work that goes into a lot of the things I think we in Silicon Valley take for granted, just because we aren't exposed to them a lot. How much therefore those companies can also benefit from the technical abilities of Silicon Valley, or the greater – that Silicon Valley as the word, not necessarily the place.

Anyway, so I think the interesting data point there would be that most of our users and customers have never heard of a lot of these fancy no code, whatever things. They have a specific problem in their work that is really important to them that they are just trying to solve. They find us to solve that problem and we're increasingly – yeah, every time we help a company who previously was doing something manually, now do something in a more automated, highly-

levered way. As a result, either the people who were building this problem flows start getting promoted and doing more interesting things, where the company itself starts growing. There's just awesome stories that we – users and the customers rave about and we love to hear.

I think there's one whole signal data there. I think the other answer to your question is I think this has to happen. What I mean there is if you'll humor me building a little bit more and I think there's a little bit more grandiose way. I think that productivity leverage spectrum I was laying out earlier, if you start to think about how market forces apply that whole productivity leverage spectrum, gets to this interesting way of looking at the world, where the computer revolution, or the digital revolution, or whatever you want to call that was promised starting 20, 30, 40 whatever maybe more years ago, as this way to fundamentally change the way that everybody works and give everybody this access to higher levered work and computers are going to take the boring, mundane jobs and maybe some people think that it's a really good thing that automation is going to help and some people are scared of automation, because it's going to replace jobs.

Regardless of that, the benefits of this digital revolution have really only accrued to there's a very small slice of people who maybe live in the Silicon Valley in New York and whatever that work, like we're talking about. I think one version of the future is those people have become – have gotten so many of those benefits that is not really hard to compete with the googles, or the Facebooks or whatever of the world, if you're not the Google or Facebook, or high-tech company of the world.

I think the bad version of the future looks like, those people in those companies continue to race off with most of the benefit and everybody else being left behind. I think you get to some really weird political and sociological problems that are maybe some of the things that we see today in the world. I think the good version of the future looks like letting the rollout of this digital revolution continue and everybody get access to the benefits. I think without intervention, with just letting everything be the way it is today, we're not on track for that.

I think the reason I am doing what I'm doing and I'm so excited about this broader set of tools, regardless if they're a competitor of ours, or just another [inaudible 0:37:32.0] on our space, I think all of – I'm excited about all of these companies, because I think it's critical to the future of

us as humans that everybody get access to these types of capabilities and join that group that is today much too small.

[0:37:45.4] JM: My understanding is that some of your impetus for starting Parabola was from your work at Deloitte. Deloitte is a consulting company that works with a lot of large companies. I imagine that when you were inside Deloitte, you got an up-close view of the pains that some large companies have when trying to adopt modern technology. What are the most acute pain points of manual processes that you saw in large enterprises?

[0:38:22.6] AY: Well, I think the largest pain point is probably true of any process, not just the specific really manual processes, but they're human problems of how do you take a really, really boring routine process that I think most people would call drudgery, and inspire people to really do it and become really good at it and take pride in doing it. The amount of people I met throughout the companies I was consulting for, who had come up with just the most interesting mission-critical processes for their company was astounding.

From everywhere in the organization, down to I think the lowest level employee, people have – all of these people have processes that they themselves take a lot of pride in and are frequently ingenious processes that help a whole company run. The person who is creating those processes then gets stuck doing the process every single day, where they came up with this ingenious thing that's going to make the company work better, but they can only create one or two of those things because now they just have to be stuck doing it themselves.

How much more interesting if the person who could come up with an ingenious process could create it, offload it to a computer to now do it for them and move on to creating the next ingenious process that will help make their job even better. I think the place I saw this go wrong, I guess in consulting is those people – a lot of those companies would lose the people who created these really interesting processes, so that Craig from Craigslist we were talking about earlier, because they don't want to be stuck just running the process. They're really good at creating these new interesting things.

Yeah, I think really interesting and valuable to enable those people to take that inner creativity and analytical capability and express it and get out into the world and not just leave it locked within this is one or two things that they're currently doing.

[0:40:14.3] JM: Describe the software stack of Parabola.

[0:40:16.9] AY: Yeah. We've been describing how Parabola can help people flows this whole time, but to paint a picture of what that actually looks like, it all runs in a web browser. It's a very native feeling, rich app built in React that lets us have that really complex state management tree, but have everything stay in sync and up-to-date and high-performance, all those benefits that React tends to give you a modern JavaScript and React stack.

When I was first building Parabola as an initial prototype, I didn't want to mess around too much with how does this whole calculation engine of actually calculating the flows that people create work. I built a calculation engine that was again naive just in the browser. You would drag and drop a bunch of these steps onto your canvas. You'd connect them all up. When you either hit run, or made a change in one of them, you would kick off a – together, well but technically, a depth-first search through this tree of steps you've created, figure out what the minimal amount of change is that need to be recalculated are. Then just in the same thread in the browser, it would do a calculate on the steps and spit back out the results.

A core principle of Parabola is maintaining some of the best parts of Excel and of a spreadsheet, which are the direct manipulation and the immediate feedback that you get where if I'm in Excel and I type in a formula and I hit enter, my whole spreadsheet recalculates and I get to see all of the changes, whether they were correct or not correct. As a non-engineer, that's the way I've been trained to think, to make a change and see what happens.

Engineers on the other hand can look at maybe 60 lines of code and are trained to keep the state of each of those variables and function calls and stack in their head. Just not engineers, because I've never had exposure to that. I think it's very important to maintain that sense of direct manipulation and operating on an entire data set at a time and a few interesting other principles.

Anyway, so as we started turning that prototype into a real product that real customers were put through the paces, obviously running all of that just in the browser didn't really make sense. Data sizes are starting to get too large for people's computers and a variety of problems. The back-end of the stack now looks like this custom-built calculation engine that takes each of those steps that are being added to your React app in the browser, figures out the minimal amount of calculations needed to be made, adds those to a queue with a bit of a bunch of worker servers listening to each one knows how to calculate each of our steps.

There's a function that is called for column filters. There's a function that's called for joins. There's a function that's called for that sentiment analysis stuff. We have the fleet of worker servers you can grab those items off the stack, or off the queue, calculate the data, stream it back to the browser in real-time. As a user, I'm making a change and I see my data calculating and streaming right back just in real-time, even if it's a huge data set that my computer otherwise wouldn't really be able to handle. I still can't operate on it as if it is live on my computer.

There's a variety of really interesting tech challenges as a result of that architecture. The result is this really interesting product that for a non-technical user is immediately familiar, feels like the best parts of a spreadsheet while having the benefit of some of the software development best practices given to you as well.

[0:43:31.2] JM: What cloud products are you using?

[0:43:32.8] AY: Oh, sorry. We are using AWS for all of our services. The caching that happens, we use Redis for every time a step calculates and needs to write that data back, so we can cache it really quickly and Redis then stream that back to the browser, while I move on to calculate the next set of steps. We have a real interesting security model as well, where we're able to treat all these flows of individual components. We call them steps that you add to your flow as essentially stateless and the floors themselves are essentially stateless.

Every time something's calculating, we can be caching that data and streaming it back to the browser. At any point in time, we can know that we can recalculate one of those flows and not have any weird side effects. We keep data for a small amount of time as possible and it makes

the security conversations extremely easy, because there's no persistence of data that shouldn't be persisted.

[SPONSOR MESSAGE]

[0:44:29.0] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications. It's definitely hard to get engineers excited about maintaining those back-office applications.

Companies like DoorDash and Brex and Amazon use Retool to build custom internal tools faster. The idea is that internal tools mostly look the same. They're made out of tables and dropdowns and buttons and text inputs. Retool gives you a drag-and-drop interface, so engineers can build these internal UIs in hours, not days. They can spend more time building features that customers will see.

Retool connects to any database and API. For example, if you want to pull in data from PostgreSQL, you just write a SQL query. You drag a table onto the canvas. If you want to try out Retool, you can go to retool.com/sedaily. That's R-E-T-O-O-L.com/sedaily. You can even host Retool on-premise if you want to keep it ultra-secure.

I've heard a lot of good things about Retool from engineers who I respect. Check it out at retool.com/sedaily.

[INTERVIEW CONTINUED]

[0:46:05.7] JM: Let's say I want to create a workflow in Parabola and I'm running an e-commerce site and let's say, the first step in the workflow is I get all the orders from the past week and it's got thousands and thousands of orders. The next step, I want to filter for orders that were over $50 and that whittles the number of orders down. I'm creating this in a drag-and-drop interface. The next step, I want to connect the e-mail addresses of those orders to my SalesForce database. I only want to keep the orders that are of people that are in my

SalesForce database. Then the last step is I want to send an e-mail to all those people who have created orders that are over $50, that are in my salesforce database, so that I can send them an e-mail that says, “Thank you for being a repeat customer. Here's a $10 gift certificate,” or something like that.

In this case, I would have four different steps. As the data is being processed across these different steps, it's being changed on your back-end, right? When that data is pulled from the database to the Shopify back-end or whatever, for those thousands and thousands of orders, let's say on the very first step, is it being pulled into memory? Is it pulled into Redis? Is it pulled into a database? I know you're not the CTO, but I'm just wondering if you – I'm just very interested how you handle the data on the back-end, while this workflow is proceeding on the front-end, to the user.

[0:47:47.3] AY: Yeah. Without giving too much secret sauce away, yeah, we cache most of that stuff in Redis as we're pulling it in and we can failover to S3 if we need to, if it ends up being a really huge data set. We're able to do some really intelligent hashing of that data that we pull in, so we can recalculate things as minimally as possible to save on calculation time. Then I'm re-fetching if there's API calls deeply nested in there.

In Parabola when you have one of those steps on your screens, or if it's a Shopify import, you can double click on it and it shows you the entire set of data at a time. We call it our result view. It's showing you how depending how big your screen is, maybe it's showing you 40, 50 rows at a time. As you scroll through, we're able to then pre-fetch a window to view coming out of that cache of the next few hundred rows of data you see and give that to you in this virtualized result for you, so that I could have 10 million rows of data coming out of one of those steps, but I'm scrubbing through it in the UI in this really nice, performant way that I can either scroll through, or search through, or whatever that is. We do have a variety of interesting work to make that possible and feel really nice and performant for the end-user.

We also have some interesting approaches to maybe what you were getting out of that question is if I'm pulling in some huge amount of data out of a database, are we storing that so that next time that flow runs, we only have to pull the next little bit of data, rather than re-pulling all of that

data. There are some pretty intelligent things we can do if we understand the underlying SQL query that's being made on a database and we can abstract some of that away from the user.

I think an interesting insight, at least still far from most of our users is when you're developing a product that is for these type of use cases we've been describing in a previous conversation and you don't have to worry about it being – I develop a product that could be built for into applications and other types of things. We can make design trade-offs based on how much data we're probably going to have in some of these use cases.

What I mean by that is a non-engineer, or an operations manager working in one of these e-commerce company is big data in that world is very different than big data as a software engineer. When an e-commerce company is pulling in SKU data, you're probably talking about maybe tens of thousands, or hundreds of thousands, or maybe a few million rows of data, which would be a very large company if they're selling that many items.

[0:50:06.5] JM: Sure. Right.

[0:50:07.5] AY: From a technical perspective, that's actually not a huge amount of data. We actually can get away with reloading a lot of this data and not worrying too much about all of that really careful performance stuff. Instead, focus on making sure that the data is always up-to-date, that everything is really easy to reason through both as a user and as a developer on our side. I think we get a few really interesting product benefits as a result of focusing on what are the real use cases people are using Parabola for.

[0:50:34.7] JM: Right. You're not necessarily catering to the back-end infrastructure challenges of somebody processing five terabytes of clickstream data and doing some advertising-based calculation off of that. I'm sure you can process some large clickstreams.

To some extent, you're working in an environment where you have an embarrassment of riches. You have really good network connectivity, you have relatively small data sets, you have really fast browsers these days and you have great back-end cloud infrastructure. The challenges that might have looked like big data, or large objects that you had to render, or large objects that you

had to deal with, or browser issues that you might have had to deal with in the past, a lot of these are just ironed out by the software abstractions that you have today.

[0:51:29.5] AY: Exactly. I think even more than that, we're benefiting from memory is increasingly cheap and things like V8 have made JavaScript really extremely performant, both on the front-end and a surprising amount of the detailed calculation logic on our back-end actually runs on node servers as well, so we are able to keep most of our code base in JavaScript, which I think a few years ago would have been a pretty surprising statement and now maybe is not so surprising.

Yeah, so I think as maybe an engineer at Parabola, it is an embarrassment of riches, where we're able to be extremely product obsessed and focused on what is this experience for an end-user doing this pretty well-known subset of use cases and make that experience just the absolute, most amazing, best experience for them possible. We can do that given that technical embarrassment of riches, I guess that we have on our side.

It's both a very interesting architecture that I think we've had to build, because most of the open source big data processing stuff designed for present terabytes of data doesn't have the live, direct manipulation trade-offs that I was talking about before. I think we've had to custom build a lot of this stuff, but the end result is both a really interesting developer experience on our side and a really interesting product experience for our users.

[0:52:44.3] JM: There are low code programmers who use Webflow, or Bubble as their front-end and then they use Parabola for back-end data processing. I was looking at your website and some of the blog entries, or the recipes were about people who had built something significant in Webflow, or Bubble and then they had some back-end data processing needs and use Parabola. Can you give me any use case, or anecdote of this stack?

I'm just trying to understand the full stack low code entrepreneur; the entrepreneur that does not know how to write code, but today has enough tooling to create almost anything that they can imagine through low code tools.

[0:53:38.9] AY: This whole conversation we've been talking about, maybe Parabola in isolation as used in this e-commerce context during some of these specific roles for helping people automate their manual data tasks. There's definitely those other really interesting subset of users who are doing what you described. I think this past week, we've actually seen two people use, I think one was Webflow and one was Bubble, to have top products on product times that were powered by Parabola behind the scenes, then using Webflow and Bubble as the web app layer on top of them.

One was called Press Hunt and one, I'm blanking on the name of the other one. I think this Press Hunt one is an interesting example. They were solving the problem where reporters need – I hope I'm not butchering this for them, but reporters need to interview people who are experts on certain subjects when they're writing a piece. I need to talk to someone who has recently went on this type of diet and had good results, or bad results, or whatever the type of thing is. They probably don't have every one of those people in their network. There's a variety of hacked together ways that they post those listings and say, “Hey, can anybody who has this thing contact me and I'll quote them in my article?”

This user of ours had understood this problem, I think personally, and so created a platform, essentially a marketplace where journalists can post requests, people can find those requests and possibly get paid as a result, or at least get publicity for having answered some of those questions. The data of processing of fetching some of those requests from the various places they're currently hosted, whether that's Twitter or other places, consolidating that altogether into one standardized view of data and then loading that into a Webflow, or Bubble, or one of those types of tools, CMSs essentially, so that it can be dynamically shown by their view layer, tends to be the common way those use cases get deployed, if that all made sense.

[0:55:32.2] JM: Yeah. Yeah, yeah, yeah. Those make sense. Is there a low code of github today? When you look at all these workflows that people are creating in Parabola, or Webflow, or Zapier, or whatever, is there some consolidating point that people can go to and see what's going on and examples of low code stuff?

[0:55:54.3] AY: I think there are a few fledgling versions of this. If you're familiar with them, there's a group called Makerpad that's clearly been one of the big supporters of this no code

world. They are a source for highlighting interesting no code tools, providing a bunch of tutorials and bringing together, whether it's tutorials, or live streams, or user stories, or other types of things about people creating stuff with a variety of no code tools. Or I think increasing and showing off the things that they've built.

There's a few other, I think similar platforms that are all trying to be – maybe a community hub for no code people. I think there's an inherent challenge with – github does something interesting and this is one of the places where code is both good and bad, is that all code is text-based essentially and until these visual programming tools. Regardless of whether you're writing a node application, or a React app, or a C++ game engine, or whatever you're doing, you can store all of that in github and have the same view of it.

I think today, there probably does not exist a comparable version for no code tools, though I definitely think that could be a useful activity. Internally at Parabola, we have our recipe library that we are just beginning to open up to some of our more power users and we'll soon be opening up to everybody to publish the recipes they've built in Parabola. Flows and built in Parabola, publish them as a recipe, so that other people can see what they've built, consume them, create them themselves.

Again, without giving away too much, we have some really, really exciting things planned for the next six or so months going down that road. Overall for this having no code space, there's a huge opportunity to help all of these creators share, collaborate and learn from each other and maybe reuse what each other are creating.

[0:57:44.0] JM: Is the user, the prototypical user of Parabola, do you feel it is these low code entrepreneur people, or the Shopify store that is doing really well, or have you managed to get into larger enterprises, like cutting-edge enterprise, like a Netflix, or a Airbnb, or Checker or something?

[0:58:08.2] AY: I think it depends if this question is wearing my vision for the company hat, or are in the weeds go-to market hat. I think in the very near future, but future, we would like to be serving all of those groups of people. I think anybody who today does not feel fully empowered

to build their own workflows to solve the problems that they're identifying and implement their really innovative solutions they're coming up with.

Particularly those apply to manipulating data, but even if they are coupled with another tool to build a full stack app. Any of those people who today are not fully empowered, we would like to empower and get back to what we're talk about of bringing everybody up into this ability to be operating like what today's software engineers are able to operate like.

For more practical perspective, today very much seeing a few groups of those interesting users, so I think this maker community has been extremely supportive of us and we've seen really incredible things that they're building. Definitely want to make sure that they continue to be well-taken care of. The e-commerce retail manufacturing world, we also have just absolutely incredible stories coming from our users of never having been able to having done the same thing every single day for years and years and years and finally for the first time, being able to replace that manual process and start up-leveling themselves in just what that empowering, almost emotional resonance moment is. Definitely investing a lot of our products' time and to features for them.

Then I think the most interesting learning so far of our company is that serving both of those groups really well, actually by default serve some of those bigger enterprises you're talking about, where because the use cases that we are solving are really individual in small team use cases, things that an individual feels the pain really strongly, those people are doing the same things that the absolute biggest companies, those Fortune 50 companies I used to consult forward, just as they are at a small start-up, or side project that somebody is considering turning into a startup.

By solving those other two groups really well, we have without even focusing too much on it, made our way into a few of those exciting enterprises you're talking about, ranging across the stack of there's probably some logos on our website you could see and I won't divulge too many to many customers here. The absolute same features are helpful for them. It's I think been actually a surprising learning of how strong this pain point is and how horizontally applicable it can be.

[1:00:28.9] JM: What else have you seen people build with Parabola that has surprised you?

[1:00:32.4] AY: I think the flows that we maybe geek out on internally are some of them are just really crazy, powerful, complex ones. Again, as you're adding all these steps to your screen, a lot of people will build a five or 10-step flow that's replacing one of these manual processes we're talking about and they get a lot of value out of that. On the far extreme other side, we've seen people with these crazy monstrosities of 400 or 500 steps of full-on, looking at one of these things, it's like a crazy, insane flowchart maybe mapping out a detailed program. It really looks like looking at code.

The most of the most complicated one I think I've seen is someone who built their own custom bespoke marketing automation engine, basically inside of Parabola, that was pulling in lead data out of Salesforce. It was collecting Google Forms from their AEs and SDRs, who were filling out like, “Please contact this person, or please don't contact this person because they told us to stop bugging them.” It was deduping against internal customer databases, so the next time they reach out to somebody as a lead who was actually a customer, it was enriching data about all of those leads with their job titles and other types of things. I think would score the leads based on if you're a director, you get some amount of points, and if you're a manager you get some other amount of points.

They would pick the most three interesting people from each company as a result of custom logic they're applying of whatever their current focus as a company was of how that scoring engine work would select the top three. I think would turn them into a comma-separated list of I'm talking with these three people at your company and then it would send off, or I think in this case, maybe just creating a ready to send list of e-mails with all of these customs merged tags and enriched sentences and everything added in to – add personalization into each of these e-mails.

It was basically one of the more complicated marketing automation tools you could think of built entirely inside of Parabola and entirely bespoke for this marketing team’s workflow they'd identified was a really important piece of their customer acquisition process.

[1:02:36.5] JM: Fascinating. Last question. If you aren't building Parabola, what business would you build?

[1:02:42.4] AY: I think a lot about as we evangelize productivity leverage and opportunity costs to our users and customers. I think a lot about that internally. I always want to make sure that I am working on the highest leverage, best opportunity cost thing possible. Certainly, currently find Parabola to be to that thing.

I think there are other domains of problem at the moment that touch on similar types of things. I think the interesting dynamics that overlap tend to be in this productivity inequality spectrum I was talking about, or in marshalling market forces. I have stayed away from the crypto world while that was buzzy before this no code world now maybe as buzzy. I'm really bullish on that world in general and I think that there's going to be some very interesting ways of organizing humans and helping apply and marshal market forces to human – put political, sociological problems that today are being solved by, or maybe not really solved by either a lot of delegating them to companies, or letting our current broken political system try to solve some of these problems.

I think some of the most interesting crypto projects at the moment will hopefully in the near future be able to credibly be a solution to some of society's bigger problems. I don't exactly know how it gets from fledgling idea, people like to talk about it and it gets a lot of hype phase, to actually solving real-world problems phase, but I hope that smart people will pick up some of those problems and work on them as well.

[1:04:20.9] JM: Right. It is conceivable. Alex, thanks for coming on the show. Great talking to you.

[1:04:23.9] AY: Absolutely. You as well.

[END OF INTERVIEW]

[1:04:34.5] JM: Looking for a job is painful. If you are in software and you have the skill set needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace that connects highly qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies. Access is exclusive and you can apply to find a job through Vettery by going to veterry.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters, so that you only get job opportunities that appeal to you. No more of those recruiters sending you blind messages that say they are looking for a java rock star with 35 years of experience, who's willing to relocate to Antarctica. We all know that there is a better way to find a job.

Check out veterry.com/sedaily and get a $300 signup bonus, if you accept a job through Vettery. Vettery is changing the way people get hired and the way that people hire. Check out veterry.com/sedaily and get a $300 signup bonus if you accept a job through Vettery. That’s That's V-E-T-T-E-R-Y.com/sedaily. Thank you to Vettery for being a sponsor of Software Engineering Daily.

[END]