EPISODE 832

[INTRODUCTION]

[0:00:00.3] JM: Software engineering is harder than it should be. There are many people who have an app idea that they are just not sure how to build. Some of these people who have an app idea are highly technical professionals, like real estate agents, scientists, accountants. These professionals learn to use spreadsheets in their day-to-day work. The spreadsheet is familiar to many people. Spreadsheets are used by young people such as students.

Spreadsheet users vary in terms of how familiar they are with the programmability of a spreadsheet, but there are certainly more people who have built complex spreadsheets than there are people who have built complex web applications.

Airtable is a tool for making application development easier and more accessible. The Airtable interface is similar to a spreadsheet and can be used for most spreadsheet applications. It can also serve as a rich back-end database system to improve the productivity of software developers, who are fully capable of building web applications.

In Airtable, there are high-level programmable components called blocks and there are integrations with developer API's, like Twilio and Stripe. Airtable has a permissions and collaboration system that allows interaction between engineers who might be using Airtable as a programmatic transactional database, and operations members who might need to read or edit specific parts of the data on an ad-hoc basis. This makes it quite a fresh and differentiated unified system for doing programming and all kinds of other things that need to be done within a business.

Howie Liu is the CEO of Airtable, and he joins the show to talk about his vision for the product and the engineering problems that he has worked on to realize that vision. Airtable has not been trivial to build. It has required its own custom database back-end and its own JavaScript, or rendering system, which is why this is one of the crazier interviews that we've done on Software Engineering Daily. I really enjoyed this conversation, because there's a degree of technical

depth and a degree of just product strategy that is instructive and pretty different than any other interview I've ever done.
I want to give a special thanks to Gareth Pronovost, who is a full-time Airtable expert that I found on YouTube. He was generous enough to take some time to have a call with me and describe his personal experience using Airtable. He's a highly-qualified technical professional that has shifted to becoming a full-time Airtable consultant and expert. The fact that there is a full professional around creating Airtable applications, speaks to how unique this platform is and I am just very intrigued by Airtable. I hope you enjoyed this episode as well.

[SPONSOR MESSAGE]

[0:03:18.1] JM: When a rider calls a car using a ride sharing service, there are hundreds of back-end services involved in fulfilling that request. Distributed tracing allows the developers at the ride-sharing company to see how requests travel through all the stages of the network. From the front-end layer to the application middleware to the backend core data services, distributed tracing can be used to understand how long a complex request is taking at each of these stages, so the developers can debug their complex application and improve performance issues.

LightStep is a company built around distributed tracing and modern observability. LightStep answers questions and diagnoses anomalies in mobile applications, monoliths and microservices. At lightstep.com/sedaily, you can get started with LightStep tracing and get a free t-shirt. This comfortable well-fitting t-shirt says, “Distributed tracing is fun,” which is a quote that you may find yourself saying once you are improving the latency of your multi-service requests.

LightStep allows you to analyze every transaction that your users engage in. You can measure performance where it matters and you can find the root cause of your problems. LightStep was founded by Ben Sigelman who was a previous guest on Software Engineering Daily. In that show, he talked about his early development of distributed tracing at Google. I recommend going back and giving that episode a listen if you haven't heard it.

If you want to try distributed tracing for free, you can use LightStep and get a free t-shirt. Go to lightstep.com/sedaily. Companies such as Lyft, Twilio and github all use LightStep to observe their systems and improve their product quality.

Thanks to LightStep for being a sponsor of Software Engineering Daily. You can support the show by going to lightstep.com/sedaily.

[INTERVIEW]

[0:05:29.2] JM: Howie Liu, you're the CEO of Airtable. Welcome to Software Engineering Daily.

[0:05:32.3] HL: Thanks for having me.

[0:05:33.0] JM: Yes, great to have you. Airtable came out of your top-down idea of making software engineering more accessible. You could have started lower level, you could have made a programming language, you could have started higher level, you could have played with a WYSIWYG site builder thing, like a Squarespace. How did you settle on a spreadsheet-like interface is the right place to start a low-code platform?

[0:05:56.6] HL: Yeah. We put the emphasis really on the fact that it was a relational database, happened to have a spreadsheet interface, not the other way around. I think the important distinction there is relational data is this ideal way to represent information. It's literally mathematically ideal for representing sets of whether it's contacts and companies, or cars and manufacturers. There's real-world information that has relationships with each other.

I think that if you actually look around at useful apps that are built today and not just the consumer-facing ones, like Snapchat, etc., really, like the majority of apps out there which are either internal to a company, or useful, more utilitarian apps, 99% of the value usually of the app is just the data model. It's the database. When you get a CRM, it's the fact that it's got he contacts and the companies on the deals table.

Really, I think as a sort of a pretty apparent place for us to start building this. If you think about the model-view-controller abstraction of a software development, starting with the model layer, because I think that was both intuitive to the end-user, but also where the bulk of the value for many apps is.

[0:07:05.8] JM: Model view controllers; one way you could look at it. Another way is in order to build a computer, you need storage, state management and computation. What's the computation side of Airtable?

[0:07:18.2] HL: Yeah. Part of is you, the human. I think a lot of the use cases for Airtable, the way that we see Airtable being used in the wild by our customers is to have this really great, friendly database that anybody can come in and manipulate, change the schema, add data, but then they can themselves in the real-world as this asynchronous, human-driven computation process, like do stuff with that data, right?

For instance, if you're a video production company and each row represents some scene, or shot of a production, then the computation is you go out there in the real-world and you shoot the scene, and then you come back and then you update it in the database. Then I think more literally, we introduced about a year ago, we call our blocks add-ons. Blocks are basically miniature apps that run on top of each of your end-user databases and actually do perform computation on top of it. It could be actually going and doing batch updates. It could be actually processing images and other content via AI, API, like Google Cloud Vision, and then returning the results into the records, or visualizing in a different way.

[0:08:26.9] JM: As a developer, I look at Airtable and I am almost intimidated by the amount of possibility from it. What I mean by that is I don't feel I even have the workflows ingrained. I'm just thinking in terms of microservices and continuous delivery and stand-up a thing over here and connect it to this thing over here in this workflow that I know at this point is basically inefficient. I should be excited about a low-code platform. Why am I intimidated by it?

[0:09:01.8] HL: Yeah. Well, it sounds like we've failed a little bit in our goal of making the product very delightful and accessible. I think, the goal really is for Airtable to feel like a Lego kit. As real Legos are marketed both in this generic, like you can buy the box of just all the different jumbled colors and shapes, or you can buy one of the pre-made templates, right? You can get the Star Wars, or the treehouse Lego kit, and it gives you instructions on how to assemble it.

I think depending on what you're trying to do and how much of the blank canvas you want to start with, we can give you a different, either more prescriptive, or less prescriptive experience, as we have templates, we have what we call Airtable universes, which is real templates or bases that other people have shared from their actual real-life usage of Airtable.

[0:09:52.5] JM: Let's say I want to build a photo sharing app. Tell me how to do that with low-code. I want mobile apps, I want a website.

[0:09:59.2] HL: Yeah. You probably wouldn't use Airtable for that. I mean, you could, but it would be a little bit of a stretch beyond its intended purpose. Because I think there's – if you draw the line between building software, building something like a photo sharing app, building that next Instagram, which is meant to be a massively consumer product, versus building really a software that's useful for you, your own personal hobbies, or work use cases, etc. We're squarely optimizing for the latter. It's more like, I think the CRM use case is a good example, where that's – you want to build the perfect way for your company to manage your customers and your inventory, let's say, or your sales. Your company has a slightly different way of doing that than every other company out there. That's the software that you would build with Airtable.

[0:10:47.5] JM: I could build a photo-sharing – Let's say I'm a influencer marketing company. I want to make a all-in-one influencer marketing platform combination photo sharing app. In order to do that, I need some layer that is accessible by these highly technical, but noncoding marketing people within the organization. It would be great if they could go into a spreadsheet-like interface and easily see all the activity that takes place across my photo sharing platform. Why wouldn't I just set up AWS lambdas as the backend and just – maybe I make some low-code mobile apps as the front-end and I just have AWS lambda as glue between Airtable as my back-end and photo sharing app as my front-end.

[0:11:32.5] HL: Yeah, so you could do that. In fact, I think that falls into a category of use case where Airtable is the CMS, if you will. It's got this GUI, where let's say the non-technical internal employees can come in and manage or view content, but then that content is also published in a much more massively consumer way with some other, let's say custom iPhone and Android app that you've made, right?

I think the key thing that you would need there, which people have done is to build your own caching layer. Basically, Airtable's API isn't designed to be consumed directly by millions of – depends on how successful your –

[0:12:11.2] JM: Could you make a cache in a block? Would that be a good application of a block?

[0:12:15.0] HL: Yes. In fact, I think the goal for blocks, right now, blocks can only be built by Airtable. We've built 30-ish blocks. We've from day one built them in a way where we can enable anybody else to build them. As opposed to hardcoding these blocks as just chunks of code that are glued into our main code base, we've actually developed from day one a developer platform, but just one that only we have access to to build blocks. Eventually, the goal is to enable you or anybody else to go in and build your own blocks.

Yes. The short answer is I think that would be absolutely one of the potential use cases for it. I think there are other things we could do around for instance, there's a lot of really cool stuff out there in let's say, the React ecosystem. I think, we can basically enable you to plug a lot of those React components, especially if you can deploy them not just to the web, but also via React Native to your custom mobile apps with zero effort.

Then all of that work actually can draw on arguably one of the hardest parts of an app to build is the data layer, especially if you want a real-time data layer, right? We've done all that for you. You would be able to go and just define the view layer, if you will. Then have that piece together the components, build the ones that you want and then have your own custom interface to it.

[0:13:34.2] JM: What else do you need to get the full software development lifecycle?

[0:13:37.6] HL: You mean in the bigger vision of Airtable?

[0:13:39.4] JM: Yeah. Yeah, yeah.

[0:13:40.9] HL: There's the full software development lifecycle as we know it today. I think that there's also this idea of opening up an entirely new class of software that can be built, and especially by non-developers, right? I think, if you compare Airtable one-for-one against the traditional way of actually writing code and building software, then we're not going to have in the immediate term some of the things you have in traditional development, right? You don't have necessarily, something like a version control system, right? You can't fork an Airtable, although you can duplicate a base and get this makeshift way of creating different copies and forking off of that.

[0:14:22.3] JM: You have provision history?

[0:14:23.0] HL: We do on a row level, as well as basically, there's something that is like Apple Time Machine for a base, so there's snapshots over time. We don't have this idea of you can group together a bunch of changes into a transactional commit, let's say, and then fork it. I think also, it's not necessarily the intent of Airtable to come in and one-for-one replace traditional software development.

For a large class of useful software, even ones that can be built – even if you're a software engineer and could write your own software, I think a lot of times, it's not desirable to go and step down to that low-level interaction, right? I think, what we go to –

[0:15:01.6] JM: You're talking my language, by the way.

[0:15:03.4] HL: Yeah. I mean, we have all kinds of internal use cases of Airtable. I mean, literally hundreds of Airtables that we use. As an example, one of them is to basically create our own homegrown version of a circle CI type of interface, right? You can see all the different builds, each of which is a record in Airtable, and you can do things, like set a different status on them and have them rebuild, etc. That's an example of something where we wouldn't have wanted to go and build that from scratch, but here we got to just build the connective glue, just build effectively that lambda that connects it to that actual build servers, to then be able to visualize it in this CMS layer.

I think, our goal is not to replace traditional software development one-for-one, but to create a new category of a easier software creation. I think we get away with not having every single one of those traditional pieces and version control.

[SPONSOR MESSAGE]

[0:16:07.3] JM: Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin prepares for these outages. Gremlin provides resilience as a service, using chaos engineering techniques pioneered at Netflix and Amazon.

Prepare your team for disaster by proactively testing failure scenarios. Max out CPU, black hole or slow down network traffic to a dependency, terminate processes and hosts. Each of these shows how your system reacts, allowing you to harden things before a production incident.

Check out Gremlin and get a free demo by going to gremlin.com/sedaily. That's gremlin.com/sedaily to get your free demo of how Gremlin can help you prepare with resilience as a service.

[INTERVIEW CONTINUED]

[0:17:06.6] JM: You've talked to people who have built spreadsheet products in the past. What are the fundamental engineering problems in building a spreadsheet-like interface?

[0:17:17.5] HL: Yeah. I think that we also have the additional challenge of one, being real-time collaborative, which of course, Google sheets was the pioneering innovation there. Really, they bought another company and then turned that into Google sheets. Not only that, but then also to do so on what really is a relational database structure underneath. That actually turns out to be harder than even just building a spreadsheet on the web real-time collaborative, right?

On the traditional spreadsheet side, I think one of the challenges is you have basically a user-defined dependency graph, right? You can define in a traditional spreadsheet any arbitrary cell can be a formula and the formula can reference any other cells, or rows, or columns, wholesale, and then perform some calculation obviously on it.

Effectively, what that means is that there's this user-defined dependency graph. Every time you change any given cell, that cell could cause other cells that have formulas in them to change. Those changes could propagate yet to other cells, and so you basically have to represent in an

efficient way this dependency graph and then have an efficient way of recomputing all those dependencies once one of the underlying, or origin cells source cells has changed. I think that's one hard part.
Traditionally, offline spreadsheets like Excel are basically held entirely in-memory. Basically, all operations performed are done in this single-threaded way. Although interestingly enough, this is an arcane piece of knowledge I picked up along the way, I believe Excel for Mac is single-threaded. Even if you have eight cores on your computer, you're going to wait around for that one core out of eight to be chugging through all these dependencies. Whereas on Windows, actually they've done some more optimizations, where in some cases you can actually parallelize that dependency graph traversion and optimize it for multiple cores. It actually runs a little bit faster if you have a multi-core CPU.

I think there's also a lot of the challenge of just building a performance-optimized UX. You're dealing with a highly dense interface, right? With a lot of different things going on, especially if you're building on the web. If you just did the naive thing, you would be rendering for a 10,000 by a 100 column spreadsheet, you'd be rendering a million de minimis DOM elements, and probably much more than that, because each cell would be represented by a few different DOM elements and so on. There's a lot of performance that you have to do as well.

[0:19:43.4] JM: When you started Airtable, you have this idea and you did a lot of research upfront. It seems you did a lot of architectural planning upfront. What did you have to do before you felt comfortable writing the first line of code?

[0:19:58.3] HL: Well, I think it was a little bit of – there was a little bit concurrency there, right? As in, I think that you want to do both at the same time, because I think working in the media, actually going and writing those lines of code. Obviously, you can spend a lot of time writing code that won't matter. I think picking the right code to write. In our case, we believe that we could figure out all of the hard back-end architectural challenges of creating a real-time collaborative database, right? There are some hard challenges there. You have to for instance, figure out your own way of handling merge conflicts, if two different people are editing the document at the same time, or the database at the same time. You have to do in a way that's actually more sophisticated, or more complicated than the way that Google sheets works, which is they have a much simpler data structure. It's a 2D, basically array.

They have what's called operational transforms, which allow you to then handle a very narrow effectively set of primitive operations, if they conflict with any other user during that time. With a relational database, because the data model is just more complicated; you have foreign key relationships, you have different field types, etc. All that stuff makes it harder.

All that being said, we didn't think that would be the hard part of Airtable. We figured that was a solvable problem. It wasn't this mathematically impossible problem. Actually, what we started be risking when we wrote lines of code was the UX, because the fundamental, I think observation that we had as we went and talked to people who had worked in the space; people who worked on Microsoft products in the space, or people who have worked on independent companies in the let's reinvent them spreadsheets, or let's reinvent databases type field, was actually like, that it would be very, very easy to fall short of building a database that really anyone could use. Instead, end up with a clunky, heavyweight product that has to be IT, or admin-driven, right?

There's so many products out there in the world that are databases that are admin-driven. I mean, there's an entire category that Forester's and Gartner's identifies as the low-code application platforms market, right, that you hinted at before. I think every single product in that category is basically super clunky and not something that your typical business and user and frankly, not even something that most software developers will be able to pick up in a few minutes, or even a few hours. You have to read a manual.

The biggest de-risking that we wanted to do was can we actually go and figure out the UX for this product that isn't just a spreadsheet? It's actually this full-on relational database masquerading as a spreadsheet. Get the UX so nailed, that any end-user can come in and figure it out, and so doing a lot of user studies and building prototypes first, as opposed to just dashing off and architecting the backend.

[0:22:50.1] JM: Did you have a clear vision for the block's interface even at that point?

[0:22:55.5] HL: I think we had a pretty clear idea of adding dynamic functionality to Airtable. We knew that we would start with this relational database. Then we wanted to introduce the ability for people to layer on top of it, custom interfaces, custom workflows and functionality. I don't

think we had the exact pixel perfect idea of blocks at the time. Actually, a few technological developments happen between the very starting days of Airtable in 2012-13. Then when we actually built blocks, which is for one, proliferation of React and the ecosystem there really happened. It just created this really clean – I mean, we could have done blocks without React.
[0:23:41.2] JM: Well, you did back on?

[0:23:43.1] HL: Well, blocks didn't exist before –

[0:23:45.3] JM: For the spreadsheet.

[0:23:46.1] HL: We actually used our own framework. We built our own. For a number of reasons, I think we now use React at the edges. I think that actually, there are –

[0:23:55.8] JM: You still use your own –

[0:23:56.9] HL: There are some benefits to – I mean, there are a lot of benefits to the way that we do it. For instance, I think there are some performance advantages in being able to handle the exact re-render cycle. We have optimizations that actually – there are actually ways to control the repaints and reflows and so on. You can do that stuff in React, but sometimes because React is meant more for a fire-and-forget approach of just, it figures out efficiently how to change the most efficient set of operations. We needed to actually have more control over that for a lot of the –

[0:24:30.5] JM: Do you think open source that framework? Because that sounds like something that there's a lot of applications, where especially with as web assembly proliferate and you get more these crazy, highly interactive front-end stuff?

[0:24:41.5] HL: I mean, I think there's two answers to that. One is it's pretty idiosyncratic to our needs. I think if we were to even come close to open sourcing it, I think it would actually require a ton of effort. We would basically end up creating a completely new framework that happens to be inspired by some of the concepts in there. I think –

[0:25:02.1] JM: You're blessed with many greenfield opportunities, many greenfield opportunities. Getting more concrete, less speculative, what is the underlying database for an Airtable base?

[0:25:11.4] HL: It's actually our own. If you look at for instance, Google sheets, Google Docs, there's not some open source, or off-the-shelf document engine they're using for that, right? They have to build their own. Similarly, we had to build our own, because we can't for instance – actually when we prototyped Airtable and we have more focus on the front-end before we built the back-end, I think for a while, we did have an intentionally hacky solution, where we actually literally would go and create a MySQL table that mapped one-to-one to every end-user table, right?

If you had a table called contacts, with a name, first name, last name, we would actually be running create an alter table commands underneath, to represent that as one-for-one as a MySQL table. As it turns out, we knew that that would never be the scalable solution, or the workable solution, because just as one example, if you have a long dependency graph in a database, for us, that comes through – we have a formula field that can draw on other fields, much like a spreadsheet formula. Then we also have implicit dependencies in the form of foreign key relationships, right? We don't call it foreign keys in the UI, but if you have a linked table, so contacts linked to accounts, then when you change a record on one side on the contact side, then you have to update, sometimes you have to update something on the account side, because you can have lookups and roll-ups that draw from that foreign key relationship, right?

All that is to say, if for every traversion of the dependency graph, like as we propagate a change through every node in that graph. We have to go and make another hard database query to pull the data for that cell and then update it and then write it back and then go to the next cell and so on. It would just be way, way, way too slow and expensive, right? There's the famous a list of 10 numbers. It's the latency speed, thinking the L1 cache of the processor, the L2 cache, the RAM, the hard drive, network latency, etc. Effectively, you'd be paying this massive overhead for every single one of these traversions back and forth to this MySQL database. You might have to do that a hundred times or more within a single keystroke of the end-user.

It was pretty obvious to us early on that we had to build our own effectively document engine, like what Google Docs has. Unlike Google Docs, actually supporting a full relational data model in it.

[0:27:40.4] JM: You can take anything off the shelf, or was it like, take a storage engine off the shelf or something?

[0:27:43.8] HL: No. I mean, I think there were a few learnings here. I mean one, we actually spent a long period, actually literally reading up on academic papers about how spreadsheets were implemented. We did look at the open source stuff, or the off-the-shelf things. MeteorJS and Firebase, I think RethinkDB had come out. We did a lot of research into – Michael Stonebraker had –

[0:28:05.2] JM: Yeah, VoltDB.

[0:28:06.2] HL: - VoltDB, which yeah, I don't know if it's still around.

[0:28:08.6] JM: It's still around.

[0:28:09.1] HL: Okay, cool. Generally, looked at a lot of his talks and other people's talks on the topic. Ultimately, came to the conclusion that obviously, it's a little bit of a bold that to take to build your own data store, right? Ultimately, every database, every data store makes its own trade-offs, right? Not just in the simplistic cap theorem terms, but even more idiosyncratic trade-offs has its own capabilities.

If you look at RethinkDB, they support a certain set of applications and use cases, but not others, right? Well, MemSQL same and so on. We came to this conclusion that if we were building a one-off vertical application, like if we were building just a CRM for let's say, pet shop owners, then it would not at all have made sense to make the investment into our own database engine. We probably should have used something off the shelf, even if it only had 60% of what we needed.

Given that it was pretty core to the entire business and the product, right? We also wanted to build something that the product experience would either be limited by, or enabled by the nature of this data store. Literally as I describe this dependency graph performance issue, we had to have a database that would work really, really well for the needs that we expected our end-users to have.

Also, because I think we – we're basically a vertically integrated product, right? As in, if you were going out and designing your own developer database, like Volt, or Rethink or so on, the challenge is you have to support a wide range of different developer use cases on top of it, right? For us, we have to support a wide range of end-user use cases, but we only have to support one developer on top of our own database engine, right? Which is us.

Effectively, we create the only front-end that's allowed to talk to directly to our data storage engine. We're building that end-user interface. I think we can actually make a lot more vertically integrated decisions around what types of behavior that database engine can and can't support and where we make trade-offs and so on.

[SPONSOR MESSAGE]

[0:30:33.9] JM: When I was working full-time as a software engineer, I was always trying to start projects within the companies that I was working. Those projects were often crazy. They were sometimes unrelated to the core business of the company I was working at. I assumed this would be desirable for the companies, because this is how companies disrupt themselves. This is how companies develop new products. It always struck me that that was so hard to do in any of these organizations. It was really hard to get projects started and it was really hard to find other people to work on those projects with me.

I have been thinking about that problem ever since I left the software industry to start Software Engineering Daily. Now I have a product that I'm working on to solve that problem, which is FindCollabs. FindCollabs is a place to find collaborators and to build projects. FindCollabs has an open network and it also has a closed enterprise system.

In the open network, you can find collaborators to work on your projects with from all across the world. In the closed network enterprise offering, you can just create closed projects for people within your company to collaborate on with.

All of it is free right now. You can sign up for the enterprise offering by just logging in with your corporate e-mail address, your corporate Google e-mail address, like if it's at softwareengineeringdaily.com. We run on Gmail, so we log in and we can create projects within our company.

We actually use FindCollabs to create and manage collaborative projects within Software Engineering Daily, so we are dogfooding this product. This is something that I really think lots of companies could find useful, because you have innovative engineers within your organization and they've probably got some awesome ideas. FindCollabs is a place for them to create ideas and find each other.

Also, if you're in the mood to run a hackathon, FindCollabs is a great place to run your corporate hackathon. If you have any feedback on FindCollabs, I would love to hear it. Just send me an e-mail, jeff@softwareengineeringdaily.com. This is really something I want to exist in the world and that's why I'm building it.

Thanks for being a listener to Software Engineering Daily. I hope you check out FindCollabs at findcollabs.com.

[INTERVIEW CONTINUED]

[0:33:23.3] JM: I want to take you out of the technical for a moment and go to strategic, because I think you have the biggest technical mote of perhaps any company that I have talked to, aside from an AWS or something. You're in very early days. You've built your own database, you've built your own front-end JavaScript serving layer, which absolutely makes sense for your domain specific use case. You have a design mote also, like very distinct design and the blocks is going to be an additional mote on top of it. You got motes on top of motes. How does that affect your strategy? Obviously, it's a serious matter of not dropping the ball, getting the culture right. Tell me about your strategic thinking at this moment.

[0:34:12.9] HL: Yeah. It's a great question. I think there's certainly some amount of luck, as well as hard work on behalf of the team that got us where we are. I think it was somewhat intentional. When we started working on the product and before we even did write that first line of code, I think we thought long and hard about the type of company we wanted to build, right?

I think that generally, markets are eventually efficient, right? They're eventually consistent. Eventually efficient, right? As in any sufficiently lucrative, or potent market opportunity, you're probably not going to be the only entrepreneur who comes up with the idea, or recognizes the idea. Unless, it's super vertically deep, like it's something that requires years of R&D, let's say a lot of the biotech stuff for instance, where you actually have this deep insight that nobody else has into a new type of drug, or biotech.
I think generally speaking for consumer or software, probably you're not the first one to a thought of some concept, right? Then it's a question of even if you go and build this thing, how are you going to be the one to succeed? Why you and why this company? We intentionally wanted to build a company where we could really take a slower, more deliberate approach to investing into something that just had a pretty high execution bar, right? If you build a super simple, let's say social messaging app, and obviously there's some challenges of scaling that up. It's pretty easy to build the V1 of let's say, a Twitter. I mean literally, the product was built in a weekend, right? That's not to demean the value of it and I think it's a great company today.

It means you're going to have a really, really, really stressful early time, because anybody else could literally go and spend also a weekend building it. For Airtable, it was deliberate that we picked a domain where we felt great design, and not design in a just visual, or even interaction, or even UX sense, but holistic design, which I include software engineering into that, really thinking about the problem to be solved, the human opportunity, the human problem to be solved. Then aligning all of the engineering effort, all of the actual front-end interface design effort, all of even the go-to-market strategy effort around to come up with the perfect elegant solution, even if that took us a long time, then would allow us to have a less stressful – though still pretty intense I think execution environment later.

We spent three years building this initial product before even launching. By contrast, my first company that I worked on was a year literally from start to finish. I spent three years longer, or three times longer just getting this Airtable to launch, than I did on the entirety of this last company. The reason was because we felt that the market opportunity was profoundly large and that those three years would actually buy us more competitive advantage and less stress down the road.

I think, now enables us to think longer term. I do think there's certainly going to be competition down the road, and the most viable competition will come from the larger companies; your Amazons and Microsofts and Googles. Because it's not something that anyone could replicate overnight, we can be more thoughtful in terms of designing our engineering culture and investing into hard problems that may not pan out for months, or even years.

[0:37:44.9] JM: Are you starting to place those speculative bets today?

[0:37:49.6] HL: Well, I mean, I think blocks itself is one example where we launched what you see as blocks a year ago. I think we had been working on that for over a year, at least before that even. Especially at that time as a 20, or 30-person company when we started working on it, I think it would be unheard of for most startups to go and invest in something that would take a year to launch. We really bit the bullet and said this is core to the vision and it's worth the time investment.

I think today, I would say it's actually – it's permeated into everything we do. When we think about our go-to-market, or our marketing efforts, we're really not just investing into payout that we can see in the next month, or two months, or three months. For instance, really designing and building an authentic brand for the company is something that will take years, I think to fully realize, just even start to fully realize its value from a business standpoint and yet, it's something that we're actively thinking about now.

[0:38:53.6] JM: With Amazon, we see the benefits of starting with a low margin parsimonious business. With Google, we see the benefits of starting with a money printing machine. How do you calibrate your sensibility when you could lean in either direction?

[0:39:14.3] HL: Well, I think the natural characteristics of the business are definitely higher margin than the selling commodities early days of Amazon. The product has very high gross margins in terms of server costs. We're not for instance, storing massive amounts of data and then reselling, or marking that up by a small margin. Really, the value is in the application layer and it's not dependent on the hard storage costs, or the hard computational costs underneath it.

I think, we've always tried to focus on creating as much depth of value for our customers as possible. What that means is rather than focusing on just creating a very thinly valuable product and then scaling that to as many people as possible, we want to both get breadth and depth of value. Building things like blocks, building – really investing into the functionality of the product, so that people can do more things, enables us to charge people a price point that we believe is premium compared to say a Dropbox, or Slack price point, but yet still massively less than the value that people can create with Airtable.

All that is to say, I think probably a little bit more towards the high margin side of the world. Philosophically, I think rather than trying to scrape away little bits and pieces of cost here and there, I think we want to just focus on actually being pound-wise and deepening our value proposition, so that even more people will happily pay us even more money.

[0:40:49.8] JM: You seem to have built a completely positive, some creative, cooperative type of company, but it's also a competitive business, it's also a thriving business. Is this a mentality that you would recommend for more business people to adopt, or do you think this is more to do with the domain? Is that philosophy? Is that game theory, like a luxury of building a high-margin business? Should Uber do that?

[0:41:23.0] HL: I think it's both. We definitely gravitated towards – I mean, the way that we built Airtable and even the problem domain that we selected to work on, the fact that we're working on Airtable and not just we as in me, or even the co-founding team, but really every person at the company, we've all chosen to work on Airtable, because of – partly because of those characteristics, right? Some of those.

I do think we're very fortunate to have the fundamentals and competitive dynamics as a company to be able to take this approach. I also think that we've intentionally selected this opportunity, because it has some of those characteristics, right? I think, it really comes down in practice to specifics. I think there are some things that we can't predict around competition, right? What is that going to look like and what are the competitors going to build in terms of their product value? How much more of a green field will we have?

I think that this general idea of creating a product that really tries to empower people and create value for our customers, and not for instance, replace people is very near and dear to us and always will be. It's something that was always very important to me. I don't know if I could have worked on a company where the primary goal for instance, was to replace humans on mass.

I actually had the opportunity in college to work with robotics professor, because it was a really interesting work in robotics and making advancing robotic manufacturing. Maybe it was many steps removed from actual impact on the world, but I philosophically decided not to work on that domain, just because whether or not somebody else will do it, whether or not it's inevitable, it just felt like something that I didn't want to put my energy towards, coming up with ways to basically innovate on technology that replaces people, instead of empowers people.

[0:43:21.0] JM: Developers always grow resentful of their proprietary tools. What are you going to do when that day comes for you?

[0:43:30.3] HL: Well, I think because we're not just replacing, or trying to come in and create the same value that existing development tools do, the hope is to create such a new class of software value potential that we're not trying to come in and steal away, or monopolize something that already exists out there.

[0:43:55.0] JM: I guess, people don't resent Apple really.

[0:43:56.6] HL: Exactly. I think Apple is a perfect example of a company that's vertically integrated and –

[0:44:01.1] JM: Or Photoshop, people don't really resent Adobe.

[0:44:03.8] HL: Right, right. The product value of Apple really is a function of how vertically integrated they are. They couldn't have created everything that they created if everything was modular and open source and interchangeable. I mean, that's basically like the IBM compatible

PC ecosystem, right? I think to some extent, we're taking this approach not because we want to necessarily –

[0:44:29.5] JM: To what the margins are.

[0:44:30.4] HL: - control. Yeah. I think it's also because we can actually result in a better customer experience through that vertical integration.

[0:44:37.1] JM: Yeah. You've done YC twice. If you were to build a YC competitor, where would you start?

[0:44:42.9] HL: Actually, just to clarify, Airtable was not a YC company. It's a YC just once.

[0:44:45.7] JM: It's not? Oh, okay. I'm sorry.

[0:44:48.7] HL: No worries. I don't know that I would build a YC competitor. You mean a competitor to YC as a institution? I mean, I think YC does a pretty good job of what they do. I think there was a brief period where as they really scaled up, I mean, when I did it for my prior company, it's still a very small intimate and almost like a family concept. There were 10 to 12 companies per batch, maybe a total of a 100 or 200 founders in the entire alumni network.

Very much literally, still part of the original Paul Graham family, right? He was still the primary partner involved in the program. I think that they've empirically succeeded in in scaling it up. There are certain things that have changed. I would argue for differently, if not for worse, but there are other parts that have gotten better and gotten more systematic and structured and the resources they offer are just really powerful, especially for first-time entrepreneurs. I would not bet on my own ability to compete with YC by any means. That would probably a poor investment if I were to have to invest in myself or YC.

[0:46:02.3] JM: How did developing markets shape your thinking?

[0:46:05.2] HL: Developing markets –

[0:46:06.5] JM: Merging markets.

[0:46:07.4] HL: Oh, sure. Well, I think that in general, we wanted to work on a problem that was pretty universal on a humankind level. I think there's a long way that we have to go in terms of internationalizing our product at all. I think there are major questions for us of how do we actually price the product in different markets? Where today, frankly we just haven't had the time or resources to really think about it a whole lot outside of the US and English-speaking countries. I think it's certainly an area of opportunity. I think the concept of Airtable certainly makes sense in many different settings.

[0:46:49.9] JM: Howie Liu, thanks for coming on the show.

[0:46:51.6] HL: Of course. Thank you so much for having me.

[END OF INTERVIEW]

[0:46:57.0] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source, it’s free to use and GoCD has all the features that you need for continuous delivery. You can model your deployment pipelines without installing any plugins. You can use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your cloud native project.

With GoCD on Kubernetes, you define your build workflow, you let GoCD provision and scale your infrastructure on the fly and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It’s great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you’re investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]