SACRA

EXPERT INTERVIEW

**UPDATED**
02/08/2023

# Abhishek Nayak, CEO of Appsmith, on building an open source internal tool builder


**TEAM**

Jan-Erik Asplund
Co-Founder
jan@sacra.com


**DISCLAIMERS**


This report is for information purposes only and is not to be used or considered as an offer or the solicitation of an offer to sell or to buy or subscribe for securities or other financial instruments. Nothing in this report constitutes investment, legal, accounting or tax advice or a representation that any investment or strategy is suitable or appropriate to your individual circumstances or otherwise constitutes a personal trade recommendation to you.

This research report has been prepared solely by Sacra and should not be considered a product of any person or entity that makes such report available, if any.

Information and opinions presented in the sections of the report were obtained or derived from sources Sacra believes are reliable, but Sacra makes no representation as to their accuracy or completeness. Past performance should not be taken as an indication or guarantee of future performance, and no representation or warranty, express or implied, is made regarding future performance. Information, opinions and estimates contained in this report reflect a determination at its original date of publication by Sacra and are subject to change without notice.

Sacra accepts no liability for loss arising from the use of the material presented in this report, except that this exclusion of liability does not apply to the extent that liability arises under specific statutes or regulations applicable to Sacra. Sacra may have issued, and may in the future issue, other reports that are inconsistent with, and reach different conclusions from, the information presented in this report. Those reports reflect different assumptions, views and analytical methods of the analysts who prepared them and Sacra is under no obligation to ensure that such other reports are brought to the attention of any recipient of this report.

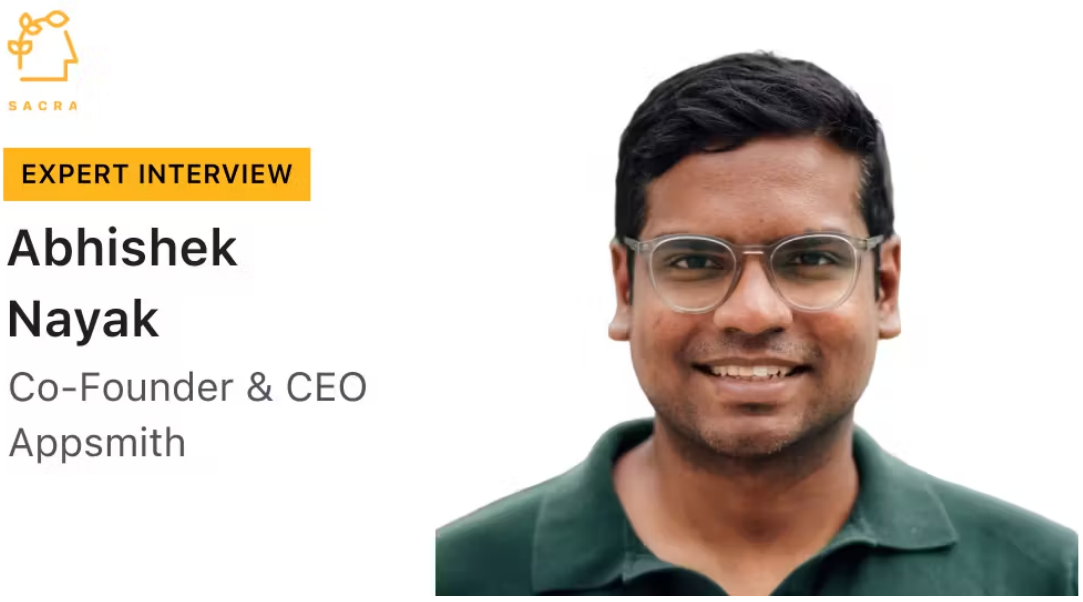
All rights reserved. All material presented in this report, unless specifically indicated otherwise is under copyright to Sacra. Sacra reserves any and all intellectual property rights in the report. All trademarks, service marks and logos used in this report are trademarks or service marks or registered trademarks or service marks of Sacra. Any modification, copying, displaying, distributing, transmitting, publishing, licensing, creating derivative works from, or selling any report is strictly prohibited. None of the material, nor its content, nor any copy of it, may be altered in any way, transmitted to, copied or distributed to any other party, without the prior express written permission of Sacra. Any unauthorized duplication, redistribution or disclosure of this report will result in prosecution.

SACRA

Published on **Feb 08th, 2023**

# Abhishek Nayak, CEO of Appsmith, on building an open source internal tool builder

By **Rohit Kaul**




## Background

Abhishek Nayak is the CEO and co-founder of Appsmith. We talked to Abhishek to learn more about the competitive landscape of internal tool builder companies and why Appsmith chose to differentiate with an open source approach.

## Interview

**Tell us about what inspired you to start Appsmith. What was your thesis behind it?**

I'm part of Appsmith's founding team of three. Appsmith was originally started by my co-founder and CTO Arpit, who's been a backend engineer for a really long time. In fact, we co-founded two startups together, before this.

Pre-Appsmith, Arpit was working at CureFit, and I was an EIR at Accel Partners. Arpit from his previous startup experiences

had seen that, at every startup, he was building a lot of admin panels, control panels, and a lot of tools to basically display some database data and interact with that. As a backend engineer, he had always struggled building the UIs for these kinds of database applications because he didn't like working on HTML/CSS.
He was tinkering around with this idea of how you might build these UIs faster than using HTML and CSS, and yet more flexibly than using something like Django Admin. This was all happening before this whole low-code/no code space emerged.

The company really got started when I was helping him out with some research about this, and I realized there was a huge space in between simple CRUD apps and the heavy enterprise apps that SAP or Salesforce sell.

I realized that this was a nice way of entering a market—by targeting developers and those who have not been buying traditional low-code products.

There are a few main differentiators that we were focused on day one and that we continue to focus on today.

The first is that Appsmith is open source.

Second, you can self-host Appsmith on your servers, and Appsmith was the first product like this where that was possible.

Now, of course, a lot of players in the industry have adopted the self-hosted model as well, but back then, there was literally no option if you had to self-host.

Next, we had noticed the other app builders—companies like OutSystems, Mendix, Power Apps, Google AppSheet and all of these players—were quite focused on the semi-technical or the non-technical user. There was nothing that was focused on the developer.

There are very specific things that developers need that a business user or a semi-technical user doesn't care about.

A developer wants flexibility to extend the platform using code. They want the ability to write code instead of clicking

dropdowns to configure something or create logic. They want to be able to write and maintain the code along with any applications that are built so that in future, it's easy for a different team member to either pick it up or extend that. That's another area where we realized, “Hey! Here's a new chance of differentiation.” Focus on the developer as the main user and the engineering team as the buyer for a product like Appsmith versus looking at traditional companies like OutSystems and some of these other low code players. They have focused on the CIO or the business head to buy their platform and not the engineer. It's a top-down motion and not a bottoms-up adoption.

These were sort of the multiple things that convinced us to start. It was definitely exciting to be the first open source player in this entire category because we knew that any category where you're building something for developers, is ripe for disruption by an open source product.

We have seen that with web frameworks, CMSs, and databases in the 2000s. That's something that we knew had to happen even for these low code application builders. We're very clear that open source is going to be a disruptive innovator to all of these other proprietary players.

**Appsmith is differentiated in the space with its open source approach. Why did you choose the open-source model for Appsmith and can you share how you think about the role that open source plays in internal tools SaaS?**

The main reason why we decided to do open source was that we realized people needed something they could host on their own servers. That was really essential, because for an application builder like Appsmith or other players to be successful, it needs to be easy for them to connect a tool with their PostgreSQL or MySQL database—otherwise, they cannot easily build an application.

When you have a cloud hosted service, the developer basically needs to go to the broad DB and white-list certain IPs and then, ensure that the cloud hosted provider can access their data. But this actually opens up more interfaces where there is more risk. The more you expose your DB to the internet, it simply is more likely that it's going to get attacked.

That's why we realized, “Hey! If it's open source, it communicates two things to developers. Number one, it's self-hosted. But number two, the source is available.” This means that if you run into a bug, you can actually go in and figure out, “Hey! Why does the bug arise?” If you want to extend it because you're not happy with something that exists, you want to modify it or you want to add something new to it, it's easy for you to go on and do that.

If Appsmith has a data source that's missing, that's not a limitation for our users because they just go on and contribute a data source. If they have a widget missing, they just go on and contribute a widget. It's like it's giving power to that engineer versus when it's proprietary, you're taking a lot of control away from that user.

Proprietary or open source, it wouldn't matter if it was a business user. If the user was a non-technical user, I don't think they would care that much about proprietary versus open source, but as soon as we're focused on developers and the technical users as our audience, they immediately identify the advantages that come with something that's open source.

**When you started Appsmith, who were your first customers? What did they use Appsmith for? How did you find your initial product-market fit?**

All our first customers and users actually came after we wrote a blog post announcing that the project was launched. In our first week, we had users from 30 countries. So, we launched the project and it just took off. None of these early customers or users were people that we knew of. We hadn't spoken to them before.

The early use cases were simple CRUD applications where somebody was displaying data from their database in a table and then providing a form to edit and update that data or add new data. The early applications were very simple because the platform itself was quite simplistic. It would let you do simple tables and forms. We also had charts so, basically these three things were the most common use cases of Appsmith.

Over time, we realized that the use cases could focus on customer related operations—customer onboarding and customer support. It could be you're running a marketing

campaign and you need a way of generating coupon codes, it could be something ad hoc like that or could be something like you're a fintech startup and you're doing a KYC process for your customers, which is an onboarding process, but it's one which has multiple people and multiple steps involved. So, the majority of our use cases, and by majority, I would say about 60% of our use cases, are related to customer operations. That's basically our focus.
The other 40%, tends to be a long tail. We have a lot of users who just build things for their engineering processes. They build UI for their CI/CD pipeline for something that helps the DevOps team, something that helps them generate sales demos, mock data for sales demos, and all sorts of long tail use cases. The long tail is really long because we have users who are in the automotive industry, healthcare, fintech startups, manufacturing industry, fire departments, universities, schools, and all sorts of companies that I wouldn't have ever accounted for in our early target total addressable market calculations and stamp calculations.

It's a really long tail of use cases that we end up seeing, but the bulk of use cases tend to be around customer operations.

**Were the people who came after reading your launch post, individual developers who are building something quick for their own teams and their orgs? Have those profiles changed over the years?**

Not really, in the sense that the core user who adopts us is always an engineer. What has changed though is the size of the teams and the companies they work for.

In the early days, it really used to be small agencies, very small startups who were adopting us. But over time, as the platform became better, we started getting adopted by mid-sized companies with up to 5,000 people.

Today of course, we have a lot of Fortune 100, Fortune 500 companies using us where people will have thousands of users on Appsmith. So, as the platform has matured, we've started seeing very, very large companies adopt us.

But the individual who adopts us, that hasn't really changed. That has always been a developer and in most cases, it's been a backend developer.

**Building and selling in India vs. the US. What does your geographical segmentation of customers look like right now?**

In the early days, a majority of our users used to come from Europe because European companies are quite sensitive about GDPR regulations, and all the cloud services that have access to their data.

Over time today, our biggest market is in the US. That's our biggest base. Our second base is India. The third, fourth, and the fifth are the UK, Germany, France, and Japan. Actually, China would be our third-biggest after the US, and India. China, would be the third and then, the UK, and some of these other European countries.

**Do you see any differences in terms of the developer requirements or the nature of developers between the US and India? If you do see those differences, how are you straddling them?**

The biggest difference that we see is, our product is a lot more quickly adopted inside a company in the US versus in India where the cost of the average developer is still low. People are happy to throw more developers at the problem instead of using an automation tool.

Appsmith ultimately is an automation and productivity tool for a developer so, when it comes to actual paying customers, we don't see those many in India versus the US.

But when it comes to open source users, we actually see quite a few users in India, but we don't see very large companies adopting us in India, yet. We see some large startups—Swiggy, Dunzo, and lots of these very well-known ones that use us today, but they generally tend to have maybe 300-400 people who use Appsmith versus in the US, where we see thousands of people inside a company using us. So there's a scale difference.

**Talking about the US, we've seen some very large companies emerge out of open source there, such as GitLab and Docker. In India, the typically large open source SaaS companies have been few—Zoho, FreshWorks, and Whatfix, all which are large SaaS but**

**proper. Do you see any blockers in terms of India building large open source SaaS companies?**
No, not so much. I think the main reason why there weren't so many open source companies—I can think of Hasura as being one which is quite popular and ubiquitous—was that people in India didn't have an understanding of open source business models.

In fact, we had trouble raising our seed round in India because we were raising it in India. People just didn't get it. It took a long time for people to understand it. Now of course you see a lot of open source players actually emerge mostly because of Hasura, Appsmith succeeding.

But the other thing that has always been true is that India has had a large base of open source contributors. There've been a lot of engineers who've been contributing to PostgreSQL, MySQL, and those large open source projects for a long time.

There's also this language called Julia Lang, which is a competitor to R that actually began in Bangalore because it was created by somebody who was a CTO of the UIDAI project. That person ended up creating Julia Lang. When you look at it, you won't even know that that was started in Bellandur in Bangalore because it's such a popular and ubiquitous project like that.

India has always had contributors, but when it came to financing, that was difficult. When it came to understanding the business model, that was difficult for local go-to-market people. That's something that is changing.

That, I would say, was our biggest hurdle but because of that, a lot of our go-to-market folks are either based in Europe or in the US where there is more awareness of these open source business models.

**What are your thoughts about the Indian SaaS ecosystem? It’s evolved over the last 8 or 10 years. What are the drivers of its evolution? Where do you think it's going?**

Indian SaaS, when it first started, was very focused on the SMB market. That's where our biggest successes are. The reason Indian SaaS succeeds with SMB markets globally, and

not just in the US, is because we are able to produce really high quality software and sell that for a lower cost base. If you notice, American SaaS companies tend to be mostly unprofitable. That's actually not the case for Indian SaaS companies. Look at Zoho or FreshWorks, and their profitability metrics are really good. FreshWorks of course, is not profitable, but they just don't spend that much on sales and marketing versus their American competitors. They just don't lose that much money. So that's a huge advantage in India that we have an amazing set of product managers, engineers who are able to build really high quality products at prices that make sense for the world outside of the US.

Today, I believe, the global software market is really big and the global software market is where India has a huge, huge opportunity. Three or four years ago, basically 70 cents out of every dollar that used to be spent on software used to come from the US. That's no longer the case. Today, that's becoming closer to 55-60 cents and we know it's going to keep going down. Some of the biggest well-known SaaS players like Notion make way more revenue outside the US than they do in the US. So that's where I see India's huge advantage.

Over the last 10 years what has happened is, the Indians SaaS companies who were pioneers, have created a base of talent which is very skilled and equipped at go-to-market design, product, and that is becoming a huge advantage for us. We benefit a lot from ex-Zoho, ex-FreshWorks employees who figured out, “Hey! This is how you go sell. This is how you go to market.” That has been helping us. I believe India because of its talent advantage and the fact that we are able to deliver a mass market product at a mass market price, will have a huge advantage.

The biggest consumer brands in the world actually tend to be mass market brands. They're not high-end premium brands and that same thing is happening for software as well. You need mass market products which can be adopted by schools and people who might have smaller budgets. But American companies cannot service these markets which do not have a large budget. That's where we have a huge advantage.

**A few years ago, VCs wouldn’t invest in Indian SaaS companies that did not have customer bases in the US because they thought that was where the largest market**

**was. Has that changed? Is it still the same? Is it somewhere in between?**
I still think it's similar in the sense that if you are not making a lot of your revenue from the world's biggest software market, which is the US, you don't really have that much of a chance to succeed.

The other thing is, for a lot of industry verticals, the US has the most sophisticated buyers and therefore, you can create the most sophisticated product because you have sophisticated buyers who are doing certain things. Therefore, the US is a great place where your product can actually improve a lot. It's like a crucible where it's just going to go through a lot more innovation than it would if you were just focused on India.

I believe a SaaS company starting from India has to be global from day one. They cannot focus on India and then, pivot and start focusing on other markets. You absolutely have to be global from day one. And that's something we believe at Appsmith because that's how we started.

Appsmith today has around 10,000 companies that use us every month, but we have sent zero cold emails and we have never run ads. It's been completely organic, which is really surprising to people because people assume that if you have to run a SaaS business, you have to go do cold email and you have to do a sales-led approach. But here’s me sitting in my bedroom, and our product reaches 180 countries of the world and 10,000 teams in a month. That is only possible today.

**When you say you're building a global internal tool for engineers, one of the top names which come to mind is Retool. How do you position Appsmith specifically vs. Retool and more generally vs. players like Airplane, Superblocks and some of the others? Are there specific use cases or industries where you think Appsmith is better suited?**

One of Appsmith’s biggest competitors is actually React.

In most places where we are adopted, people are moving away from a React or an angular project, and then, using Appsmith. The beauty of using Appsmith is you can embed Appsmith in any of these legacy projects as well. So you have a React project that's running, but you cannot extend it for

whatever reason. You can however, build it in Appsmith and simply embed that application inside your React project. This is something that's a huge advantage for Appsmith because you're able to do this in the open source and the free edition, and extend it.
The second thing that's different about Appsmith is, when you begin paying for Appsmith, it's usage-based pricing. You pay 40 cents per user per hour. You're not paying a seat-based pricing. The reason is, because Appsmith's use cases are immense and they're used by very, very large teams. What seat-based pricing does is it forces you to ensure that you are only paying for those users who are getting similar amounts of value out of the internal apps that you are building.

In case of Appsmith, some users may be using the internal apps that are built like 15, 20 hours in a month. In other cases they might just be using it for 1 or 2 hours in a month, but we are able to serve all of these different use cases because of our usage-based pricing. So that's a big differentiator.

The third one of course, which I've been speaking about is open source, self-hosted. The fact that there's a community, and that you can modify and contribute, leads to a lot of differentiation.In fact, Appsmith's project has enabled Superblocks to be built. Superblocks is essentially a fork of Appsmith, it's been built on top of Appsmith and they've gone on to raise, I believe, $40 million. They're proprietary, but they're built on top of Appsmith. So, whenever Appsmith's project gets better, Superblocks also gets better because it's a fork. About Superblocks, we don't really worry much about, because our core project I believe, is still going to be better than using a fork of it.

When it comes to airplane.dev, their pitch is something different. Their pitch is a lot more around Cron Jobs and being a developer tool rather than something that's actually an application builder that's going to be used by, for example, your customer support team or your ops marketing team. It's more of a developer tool.

We sort of get bucketed in these categories for sure by analysts and the industry, but we don't actually run into these companies at all when customers compare us.

When it comes to Retool, I think our positioning is very clear. You have to choose something that's open source so that you're not locked into a proprietary player. Use something that has usage-based pricing instead of Retool's very expensive seat-based pricing. In fact, in their self-hosted edition, you have to pay a large platform fee and only then, do you adopt it. So, something that Retool might charge you hundreds of thousands of dollars for, Appsmith gives it away for free.

We are very well positioned against them and we have a lot of Retool customers—their older customers and very large customers—who actually moved to Appsmith. Over this and the next couple of years, as our project gets more mature, we believe that's going to be a huge trend.

**I am looking at your pricing page. Don't you have a mix of seat-based plus usage-based pricing, like $0.40 plus $20 for the core user? Could you help us understand the pricing model a little bit better?**

The way our pricing works is basically it's 40 cents per hour per user, but it's capped at $20 per user per month. So for anybody who uses it a lot, it gets capped at $20 per user per month. But for anybody who uses it very little, you're just going to pay according to usage. This is how it works.

**Let’s turn to the kind of use cases for Retool vs. Appsmith vs. Airplane. There are certain platforms engineers use to build internal tools where the employee/CTO/finance team is hitting a broad database. There are other tools used to build an external tool where someone from outside the company's hitting the broad database. How different are these tools? Do your customers use them interchangeably?**

I’d say, the bulk of our users tend to be what are known as internal customers. It's an employee, could be a partner, or could be a consultant that you work with, but these are internal customers. These are not customers who you're charging money from. That's the bulk of our use cases.

When it comes to external use cases, we do have a lot of people using our community edition. We also have a few people who pay for a business edition for customer-facing use cases. But that's not something that we are well equipped for.

You can still use it, but because that's not a focus of our platform, there are lots of features that are missing in the platform that would enable that.
Over time, we do believe there is going to be a market for external-facing use cases and we'll definitely try to play in that market once we begin to see more signs for it because, I believe, there needs to be a Shopify equivalent for SaaS.

Just like every ecommerce company used to be handcrafted, every ecommerce website used to be hand-built and handcrafted till Shopify came in and said, "Hey! You don't really need to do that. There's lots of common use cases here."

That's something I believe will happen for SaaS as SaaS gets commoditized. That's again something that we will definitely play in that market. But as of today, we've not seen crazy adoption there to warrant us to focus there, but maybe in a couple of years.

**Appsmith and Retool cater to developers who can write JavaScript, React, SQL. Webflow, Airtable and GPay are for the completely non-technical folk. Could you highlight the differences in features of external tools and tools being used to build apps for internal customers?**

With tools used for building external customer-facing things, their visual design, visual components, and the flexibility that they allow is a lot higher than what is possible in Appsmith and Retool.

Let's take Webflow, Wix, WordPress for example. The visual design possibilities that are there are very, very high. The limitation is they generally don't do well with large amounts of data. They're generally static sites, used for displaying data, and not to let the customer modify or update data or do workflows. So that's the trade-off that they have to make as an external-facing product.

The other thing is, as an external-facing product, you need to provide a lot more customizability, which helps the developer control the entire end-to-end customer experience. Starting from the login page to what emails go out, when do they go out, what are all the marketing automation that you need to

build in—all of those things are necessary in those customer-facing products.
In case of internal facing products, at Appsmith, we are a more opinionated platform where we've said, “Hey! The login page, this is what it should look like. You can change the color, but you can't really change everything there that you would like.” The marketing automation that exists for communicating with people—that's a fixed thing—and you can't really modify it too much. There's some things you can change but not that much because developers don't really care about it. So those are basically the trade-offs.

The trade-off is if you want more visual customization and flexibility, you cannot have that much flexibility on the data side of things. In case of Webflow or Airtable, you cannot connect to any other database, but whatever they provide. There is no option for you to go do that. Connect, talk to a PostgreSQL or a MySQL, or if you do do that, it's very cumbersome for you to make that happen.

In case of Appsmith, you can really connect to or talk to multiple, 15 plus databases, any SaaS product out there in the world which has an API. There's nothing that's stopping you doing it.

So we focus a lot on flexibility on the data side of things and we focus less on flexibility on the UI side of things.

Lastly, data security, user authentication, and role based access controls are important features for internal customers, but not so critical for external customers.

**You said React was probably your biggest competitor. In that context, do you think you could phrase Appsmith as a lightweight PaaS used for deploying internal tools? If you agree, do you see more modern PaaS like Vercel or Netlify being your competitors in this market?**

We don't consider ourselves PaaS. I would say we consider ourselves more a custom web framework. We are a web framework that's best for building CRUD applications and database related applications.

But when it comes to PaaS, they generally tend to be more on the infra and the backend sides of things. In the case of

Appsmith, we don't give you a database from scratch, we expect you to already have a database. Those are some of the differences.
When it comes to something like Vercel, for example, we use Vercel internally for hosting our deploy previews. I can imagine people using Vercel to host Appsmith and to use Appsmith. Those things are possible. But I don't expect something like Vercel or Next.js to be a big competitor for us at least today, because there's just so many things that they are focused on, which is for the external customer-facing application. They need to pivot, do more things, and focus on the internal customer. They can certainly compete, but it's not going to be the best for them.

**You mentioned usage-based, and being capped per user. How does your revenue expand once you're inside an org? Since you're selling to the engineers, does your revenue get capped at the number of engineers an org can have?**

There's a couple of things.

First, we charge for the business users who use applications built on Appsmith and not for the engineers. So, once you build something for customer support, we only charge for the business users who are actually using that application. We do charge a little for the developer when they're building, but that's a very small percentage in comparison to what we charge for actual usage.

The idea behind the usage-based model is that it's a shared risk model. We only charge you if the application is actually used and successful. You're not getting charged just because you adopted a platform. That's something that we've heard repeatedly from people who've bought proprietary platforms where they're like, "Hey! I signed up for 500 seats, but really 50 of them or 100 of them only use it. Everything else, we hoped that they would adopt it, but they haven't adopted it much."

Again, with internal applications, you have to drive that internal adoption. If you're choosing a platform whose incentives are not aligned with increasing your adoption internally, you are really going to struggle. That's one of the reasons why with the usage-based model, it's a shared risk model. We don't get paid

if the applications are not getting used. That's amazing for our customers.
Second, the engineers and engineering teams are used to paying for things on a usage-based model. That's how they pay for infra, and the multiple analytics services. Appsmith is different in the sense that we are sort of the first application company which is charging for something like this in a usage-based model.

But that has actually worked really, really well for us and helped us get adoption in very, very large companies. These are companies who would not have adopted a proprietary platform because they might have like 3,000-4,000 employees and they don't want to pay a per seat price for every single employee there. This versus when you have a usage-based model, and they only pay once somebody begins using it and only according to how much ever they use us.

**What does success look like for Appsmith? Say five years from now, what does Appsmith become if everything goes well?**

Appsmith should have the world's largest collection of internal applications that anybody can use, fork, and get started working on.

Instead of having to build from scratch, they're actually able to reuse other people's work. That's probably what success looks like—a whole “App Store” like marketplace just focused on business applications. Something like what only exists in the Salesforce ecosystem today.

If you're a small municipality in a small European city and you need certain tools, you either have to buy them from a large company like SAP and do a big adoption, or you have to build them from scratch. There is no middle ground.

There should be tools that you can just simply download and begin using for your business. That's something that should be possible and that's what success would look like for us.

**Lastly, is there anything you’d like to add that we have not touched upon about Appsmith or the internal tools market?**

The one thing that I would like to add is about how open source is a way to commoditize the industry and a strong disruptive innovation.

First, what do I mean by commoditize? In the internal app space, there are a bunch of new players and at least three of them have actually used Appsmith's code to their businesses on top of it. What we've done is what traditional, old, proprietary players used to charge a bomb for. We've said, “That's commoditized. You’ve got to innovate.” It's forced every single player in this business to innovate in many, many different ways. By commoditizing the space, we are actually making the basic stuff available for free and forcing every player to innovate, which is great for customers.

Second, disruptive innovation. I mean it in the sense Clayton Christensen uses it, which is you have a lower cost, lower quality innovation that enters a market from the bottom and begins to eat up all those use cases, which a previous player cannot service profitably and therefore, does not want to service it. They keep going up market but ultimately, what ends up happening is that, that low cost player begins to improve the quality of product, the use cases that it captures, and it actually becomes a serious threat for all the other proprietary players that existed before this.

Those are two really interesting things that I find fascinating about open source. That's something I can see is playing out in this space really well.

## Disclaimers

*This transcript is for information purposes only and does not constitute advice of any type or trade recommendation and should not form the basis of any investment decision. Sacra accepts no liability for the transcript or for any errors, omissions or inaccuracies in respect of it. The views of the experts expressed in the transcript are those of the experts and they are not endorsed by, nor do they represent the opinion of Sacra. Sacra reserves all copyright, intellectual property rights in the transcript. Any modification, copying, displaying, distributing, transmitting, publishing, licensing, creating derivative works from, or selling any transcript is strictly prohibited.*