Zero-Shot Learning - Episode 2: Tom Occhino from Vercel AI is rewriting who gets to build

Hosts

- **Nancy Wang** - CTO, 1Password
- **Dev Tagare** - Sr. Director, Head of Engineering, Gemini Enterprise & Business, Google *(personal capacity)*

Guest

- **Tom Occhino** - Chief Product Officer, Vercel

Opening

Nancy Wang: Hey, everybody. I'm Nancy Wang. I'm the Chief Technology Officer at 1Password. Welcome to Zero-Shot Learning, which is a podcast about the reality of developing with AI from the people actually doing the work. I host today's show and this series alongside Dev Tagare, who is the Senior Director and Head of Engineering for Google Gemini Enterprise and Business.

Today's episode begins with a question about platform shifts. Tom Occhino helped shape one of the biggest shifts as part of the team behind React at Facebook. And now as the Chief Product Officer of Vercel, he's helping define how software is being built with AI. The move to component-based interfaces with React really changed how developers structured software, but the shift to AI-generated software really changes something even deeper - who gets to build, how fast they can build, and what the job of building even looks like in this age.

In today's conversation with Tom, we explore what lies beneath that shift: the tooling, the latency, observability, security guardrails, and what it takes to make those systems actually usable in production. Then we get to see it. Tom is going to show us exactly what it looks like with a real demo of V0, building a real app in real time. This is going to be a great conversation about how software is changing - but also what stays the same: taste, judgment, ownership, and really the ability to turn an idea into something that people can actually use. So lock in. This is Zero-Shot Learning.

Episode 2

Nancy Wang: Tom, thanks for joining us in our studio today.

Tom Occhino: Thank you.

Dev Tagare: Tom, it's great to have you here. You've been pushing the edge of frontend and AI and developer productivity in general for several years. So Tom, looking back at the Facebook era - specifically the shift from MVC to component-based architectures with React - does the current shift to AI-driven interfaces feel more similar, or does it feel super disruptive?

1. From React to AI: A Fundamentally Different Shift

Tom Occhino: This is fundamentally different. It's worth providing some context about the shift. We went from MVC to React - this single-directional data flow and component-based view architecture - but it was fairly incremental. Basically what was happening at the time, as we were building these sophisticated apps that had data backing them: the data changes, right? Either the user does something that changes the data, or change comes from the back end. And then we had to manually update our views so that the user could see the latest version, so that the models stay in sync with our data - and I mean models from the MVC era. It was just a manual, error-prone process, and software was kind of littered with bugs. The biggest thing that happened for us was that our software was just hard to maintain, because you have lots of different things that can change the data, and then lots of different ways of updating your views. As our teams got bigger and our code bases got bigger, things got harder.

React comes along and says, hey, don't think about most of that. Here's what you should think about: you've got some data - how do you render all your views? And it's just a liberating experience. You're like, okay, I can just think about this one way, and the system will intelligently and efficiently update my user experience. It turns out that agents like to do things that way as well, right? Overwhelmingly, React is the sort of framework of choice for LLMs. I think one of the reasons is because the code you get to author doesn't have to consider a lot of these weird intermediate states and imperative manual updating of views. React is really well suited for this new world.

But the shift between me authoring all of this stuff for myself and then LLMs outputting that stuff is just a fundamentally different approach to software. There are a lot of similarities in the sense that you still have to think really clearly about requirements and outcomes that you're hoping to achieve. You have to have opinions, and you have to have taste, and you have to be able to make decisions. But in terms of the actual implementation, that ends up being dramatically easier now - the LLM can output really, really high-quality code. We'll use V0 later - we'll use it with V0 Max, which uses Opus under the hood - and you'll see the quality of the code it can output is really, really amazing. So the way you spend your time now shifts a little bit, I think. And certainly a lot more people can become empowered to build.

2. The Prototype Is the New PRD

Nancy Wang: Well, speaking about "empower to build" - I'm going to quote you on something you said, which is the prototype is the new PRD. And so obviously this brings back flashbacks, because coming from AWS, right, we were taught to write 6-Pagers - with appendices, which really didn't make it six pages, but you get the drift. So when that becomes the new norm, how do you think teams should think about shipping product when everyone now has the ability to prototype?

Tom Occhino: Yeah. And maybe in many cases go beyond prototype. I think the idea that you would have a two-step process - this offline step where you first generate a PRD, collect feedback, iterate on it, and then send that over to the engineering team - can be a much tighter loop now. I can be generating a PRD with the agent that is about to build my app. So sometimes in V0, what I'll say is, hey, let's just plan out what we want to do here. Let's put together all my thoughts - fill in some blanks - and you can kind of fill in a bunch of content. Then you have this like intermediate PRD, but no one really needs to see it because what people really want to see is the actual product. So if you can kind of then build that, it becomes a much more effective way to communicate. And certainly you still want to document edge cases, how to handle this situation and that. And you want all of that to be codified. But one of the things we found is that V0 has actually become a communication tool as much as anything else, because you can now communicate your ideas much, much more clearly - people can play with the thing you want to build, inspect it, try things out. I think it's going to change the way that people think about building things. You no longer need to have really fundamentally separate functions that do the clear thinking and writing and then a separate team that goes and does the building. You can have kind of one team or one person that can do the whole thing end to end. And that's going to be really, really powerful.

But I also think there's a question about ownership and who's responsible now, who's accountable for things. I think in this new world, ownership is going to be as important as ever. You have to have somebody who has permission to make the changes, knows what they want to change, has taste, has customer context and feedback. Not just anybody can go in and make that app that was created by somebody else better. Sure, you can fix bugs and things. But to know strategically how you want it to behave - somebody can think much more deeply and execute on the creation of new capabilities much more quickly. I love that Vercel is building lots of the tools that you need in order to embrace this future.

Nancy Wang: Are you already seeing that shift happen in the industry for real, Tom?

Dev Tagare: In the sense that, like, I once in a while will get a fully running app from a product counterpart I work with. And it's, you know, a lot of it is mocked up, but that kind of makes sense. You interact with it, you give some feedback. Essentially, the user journey is now super interactive and I don't have to spend time visualizing and documenting out CUJs. Now it's there for you to see. And your product is obviously helping that.

Tom Occhino: So yeah, we're absolutely seeing it play out. I'd like to talk about the evolution of V0 itself, because we didn't get to that state right away. We've been building up to it. And actually the name V0 comes from this idea that we assumed this would be a tool that helped with the blank canvas problem - just like, give me my initial thing and then I'll take it from here. So initially what you would build with V0 is just a skeleton, just the UI. And then I'll go do the real software engineering.

And then with the next version of V0, it became much more conversational. I was like, oh my gosh - this is starting to build real front-end apps. And still I have a bunch of engineering work to do behind the scenes, but it's a real app. And then now we're moving into the world of this building full-stack production - like, I deployed something over the weekend that I'm using now every day. It's a real app with a real database and real auth and real blob storage, and all of this stuff was just configured and managed for me from V0.

So we've had this kind of evolution. It didn't start with the ability to capture multiple critical user journeys and all these things, but now it really does. The thing that has been fascinating as we've watched this evolution of the tool - as it's gotten more and more capable - is that the profile of people who can use it and get value out of it is just widening. It used to be a tool for developers that saved me time. Then it was a tool for developers to save me time but some other people could build some simple stuff without having to be as technical. And it's moving in the direction of many, many more people being able to get real value out of this.

I can give you some examples. I love that some of our account executives at Vercel will use V0 to present to customers instead of a spreadsheet that talks about total cost of ownership or the economic impact of using Vercel. They'll be like, "Hey, I created an app for you - a one-of-a-kind app just for this meeting that will probably never get used again - but I created an app that allows us to interactively explore things." Like, when was the last time an account executive had an engineer on hand to build an app that helps with a meeting? There's just no way - your engineering resources are far too critical.

3. Technical Deep Dive: Latency, the Edge, and TTFT

Dev Tagare: That makes a lot of sense. Switching gears a little bit into the technical deep dive section here, Tom. So specifically, I want to double-click a little bit into latency and get to what the edge really is. Coming from the Anthropic and Gemini world, TTFT - time to first token - is really the make-or-break metric in terms of your experience with an LLM, in addition to hallucinations and such. So how did you design the SDK to optimize for TTFT across different models and different runtimes, both closed-source and open source?

Tom Occhino: One of the things that we've optimized for a very long time is delivery to end users. We know that the closer you can be to the user with the majority of what you need to send them, the faster it will get there. But then there's this really interesting thing that happens where if you get to the user really quickly but then need to go to the origin to get data, or to get tokens, or to get whatever you need - you've just moved the latency. It used to be directly from the user to the origin. Now it's from the user to the edge really fast, but then the edge needs to go to the origin. So what we've done is architect a system that prioritizes getting as many things to the edge as it can, but then creating this very fast link between the edge and the origin - a very wide highway for communication and things like that.

4. Security Guardrails for AI Agents

Nancy Wang: So I have to ask - as you know, we are a security company. I have to say that your AI SDK makes it just ridiculously easy for agents to call tools. And so, putting on my security hat - and for the CISOs listening to this - how do you make sure that these agents don't get exploited, and what guardrails are you thinking about putting in place out of the box?

Tom Occhino: Yeah. I mean, under no circumstances are we encouraging code execution on the client. Of course it happens during development - you're doing some of this stuff. But the reason we created Vercel Sandbox is because we need that untrusted code execution environment that does not have access to production secrets, that does not have access to production configuration, etc. There's a really rich question of the security of agents here, and we're doing a bunch of things. One thing recently: hardening of markdown rendering to prevent data exfiltration - we actually had a blog post about this called "Building Secure AI Agents," which I encourage you to take a look at. A ton of work going on there. And then the other thing is, we created the sandbox environment, which is great because you can run untrusted code without worrying about it accessing production secrets, etc. But there are outgoing requests that come from that sandbox. So we need to be thinking about the way that agents do things from the sandbox - who are they allowed to talk to and in what capacity. It's an open area of research and development for us, but obviously something that all of us need to take very, very seriously - especially as we open up access to these tools to so many more people who don't have the security fundamentals from their prior career. We need to be creating systems that are secure by default and safe by default.

Nancy Wang: Well, it's almost like you read my mind - because one thing I've been promoting internally is you've got to make the paved path the easy path, right? Because if security gets really hard, that's where it risks becoming an afterthought.

Tom Occhino: Exactly right.

5. Is Vercel the OS for the Agentic World?

Dev Tagare: Just thinking through it - is the goal for the SDK to become the OS for the agentic world?

Tom Occhino: So we started building V0 and it's a front-end cloud app. But we needed more stuff - okay, this is a necessary but insufficient set of things. What else do we need? We needed the AI SDK because we were constantly swapping out the models in our app, because new ones are released all the time, and we needed to stop changing our product code. We needed the AI gateway because as more and more teams are using AI, they're all managing their own keys - we had no visibility into what spend was going where. It was crazy. We needed things like sandboxes, as I mentioned, for untrusted code execution. We needed a primitive called workflows because agents do this very asynchronous, workflow-like thing - wait for something, do something. So we needed to build all this new stuff, and the AI SDK became one more piece of this puzzle for us. We call the collection of these things the AI Cloud. And that's why I say - sure, Vercel could be the operating system of agents, because it's composed of everything that the agent needs in order to develop, deploy, secure, and observe the software of the future.

Nancy Wang: That makes a lot of sense - thank you for the framing, Tom. I think it ties together a lot of the building blocks in place.

6. Gen UI: Probabilistic by Design, Deterministic at Deploy

Dev Tagare: Yeah. This goes back to, Tom, when we talked about generative UI - GenUI. Because I think you're maybe on the fence about the word "probabilistic," right? So putting on my engineering hat for a moment - how do you think about this world of generative UI, where the outputs are not always the same based on what your input is? From a maintainability perspective, long-termness - how do we avoid the trap of creating tech debt, especially when core UI components are now being generated with AI? More and more front ends will be created that way.

Tom Occhino: When we coined this term Gen UI - where a model is going to be outputting UI for you -

Nancy Wang: Oh, I like that. Gen UI.

Tom Occhino: - and that UI, in theory, you could have never seen anything like it. The first versions of this - and even before AI - the way to think about it is that based on something from the user, I need to generate a user interface that I didn't necessarily anticipate exactly. If you think about Google search or Facebook newsfeed, there's an infinite number of blocks that can show up in that top box. It seems infinite - it's actually a lot. There are thousands of different types of newsfeed stories. What we did was make it so you could dynamically import code based on what the user asked for. I need to show a video story, so I'll download the code for video and display this video story. For Google search, I'd search for some calculation and it would show me a calculator. It didn't download all of the code for all the different types of things upfront - it downloaded them as needed.

So the first version of Gen UI with V0 - I think what we were assuming was going to happen is, when I have a chatbot interface now, instead of just asking for text, I can show you rich UI. And that UI had to be developed. This is like a date picker, this is a map, this is something. The LLM could invoke that and render this Gen UI thing.

But then we know what happened next - it was like, well, it's not just these deterministic list of components. It's kind of anything. And now we can generate whole UIs on the fly that have never been seen before, made of primitives and components composed together, usually with React. And what happens now is you can output UIs that are truly bespoke.

I have a bunch of mentors that have said, don't try to predict the future, just build it. So here's what I think: during development, while you're trying to learn - is this the right approach, should I put this here, should I flip them - I actually want lots of options. I want to refine and iterate, and I think it's really good that LLMs don't output the same thing every time, because it would stifle creativity. So during development, that's great. Then when we get to the actual final user experience, you want the perception of this highly dynamic UI that can really mold to the user's need - but you don't want the user to have to learn how to use the app every time they open it. So you do want familiarity and common patterns and all these other things that constrain the solution space.

And now towards where we're headed - once LLMs are outputting 50,000 tokens a second and UIs can be generated much more quickly, will it be novel to want to see a personalized version of the UI? In that world, I actually don't think it's probabilistic anymore. I think it's much more deterministic, but it's much more personalized. So I kind of move past "probabilistic" and towards either "personalized" or highly dynamic. But I think we want to know what the LLMs are outputting. And if we have users and customers, we want a predictable experience for them.

Dev Tagare: Can you imagine the support tickets that come through when you've just never seen the UI before? Right. Telemetry in this space is going to be huge - something that you all are working on in a way. But with every dynamically rendered UI, you're going to have quirks. You're going to have a lot of personalization with every single rendering, because our personalized context evolves over time. And then it's going to be insane to reproduce any moment. Which means you probably are going to ship native screen capture tools, native recording tools - which usually come with most Chromium-based browsers, in any case. But how do you stream that back, and how do you interpret it to see how it reproduces? It's going to be fun.

7. The Evolving Role of the Frontend Engineer

Nancy Wang: So as we move towards the demo section of this podcast, I'm curious - do you see the role of the frontend engineer evolving more into a system architect or a general enterprise architect who's orchestrating V0, or simply writing prompts instead of configuring CSS or putting the fine-tuning touches on things?

Tom Occhino: I think a really interesting thing happened in the early days of computer science. Every software engineer did it all - you're building the thing, so you had to figure out data storage, your algorithms, how you're fetching and retrieving, and what your UI looks like. And over time, what happened is we had to develop specializations because it was much more effective to push the state of the art forward. So we became very, very good at one discipline or the other. The most coarse-grained breakdown would be front end and back end - but in reality we have dramatically more than that: design, engineering, CSS-only front end. You have very, very specialized folks. You have back-end engineers that specialize in APIs or really efficient algorithmic data access, or even lower-level systems engineers. We had all of this specialization happen because human beings can only have so much context in their head.

So you started out in a world where everybody is kind of a generalist but the software is pretty low quality and pretty rudimentary. You accelerate towards a world where you have dramatic specialization and extreme diversity of skill set. And now I think we're moving into a world where basically everybody is about to be a generalist again - in a world where the models can output best-in-class backends and best-in-class frontends. It's going to be really about where you want to spend your energy, where you want to think and focus and iterate.

The default UI you used to get from a back-end engineer used to be pretty bad. The default UI you're going to get from a back-end engineer going forward is going to be really good. And you can certainly go further than that if you have a passion for the front end. But everybody is going to need to be able to communicate extremely clearly, which requires thinking extremely clearly, which probably requires writing because writing is thinking. And they're going to need to deeply understand how the product is supposed to behave. So front-end folks are going to get more full-stack. Back-end folks are going to get more full-stack. And everybody who has agency and passion is going to be able to create really high-quality starts.

The thing that happened with this frontend cloud thing was we kind of almost inadvertently created what I would describe as self-driving infrastructure, because the front-end engineers didn't really want to think as much about efficient infrastructure architecture and deployment - that was kind of taken care of by the system. But now the infrastructure can kind of improve itself. So that's why we call it self-driving infrastructure. I think the same thing can now happen on the front end - like, I used to specialize in how to do the exact CSS needed for that exact layout of that button. And now I don't need to spend as much time - I can spend my time where I choose to, rather than feeling like I have to go and refine this. So we're going to have this evolution from self-driving infrastructure to something like self-driving front-end development - I can just tell it where to go. It's actually quite high quality. We don't have a term for that yet, but that's the direction we're moving: everybody being able to become much more full-stack.

Dev Tagare: It makes a lot of sense. Yeah. Maybe I'll give you a term I'm thinking of, which is the "frontend Pareto" - where you reach the 95th percentile of what you want in terms of your end-to-end experience really, really fast. You decouple the infrastructure pieces and make them self-improving - not a concern of the front-end engineers anymore. And then for the last 5% you give awesome tools via your SDK and such. So you hit the frontend Pareto in like a tenth of the time you normally would.

Tom Occhino: I like it. I'm looking forward to reading the blog post.

Dev Tagare: Thank you.

8. Live Demo: Speed of Thought with V0

Nancy Wang: All right. Do we want to jump to the Speed of Thought demo?

Dev Tagare: I'm very excited. Yeah. I'm ready to jump in and build some stuff.

Tom Occhino: This is V0. Everyone, this is my personal V0 account - I'll use that because whenever we generate here, I hope it's going to need a database. I recently was talking to somebody who worked in finance, and they were used to using a Bloomberg terminal. They were like, man, I wish I could just have my version of this that had only the things I care about and got rid of some of this UI and other stuff. So I was like, let's just start iterating on a finance terminal.

I've created a prompt here - we'll read through it while it starts cooking. One of the things I want to make clear is: "make database for storing all the stocks and values." I really want to connect to an integration here. So let me just walk you through what we're going to build. This is a high-end portfolio tracking web app for a professional financial analyst. We want it to feel

like a real internal tool. The analyst was telling me it needs to be professional - some features here, some other things. The first thing I realized is, I asked it, I wanted to use a database because I think this is going to be a real production app. It's asking me if I want to install Supabase here. I've got some options, but I'm going to install it.

I just love going through this flow because right inside the Vercel marketplace, this is all ready to go. I click one button, give it a name, I'll use the default, and then I'm back into V0 and building. What's really cool here is a database is being provisioned on the back end - not a Vercel database, a Supabase database in this case. And I didn't have to go and create an account or do anything. It did it for me and now I'm back in V0. I didn't even close the tab and I can tell it to continue building. It's got what it needs to keep going. It's got a design brief, it's setting up a code base, it's going to create a list of all the things it needs to do. This seamless integration is really powerful because there's just tons of things you need when you're building an app that you kind of take for granted. I told it to keep going. It's creating some SQL tables.

Dev Tagare: Hey, let's look at our project plan. And this is super awesome, Tom, because it takes me back to one of your earlier observations, which was integration needs to be seamless. You let the database experts deal with their stuff. You just want to give a control plane to interface with it very, very natively. That's amazing.

Tom Occhino: Yeah. The thing that I love about using tools like V0 is - we talk about self-driving infrastructure for after I deploy my app, but this is kind of self-driving front-end development. And look, I was pretty detailed - I actually expressed exactly what I wanted. This is not a full PRD, but there's a lot of requirements here and it's got some details. I needed to be pretty specific about it because there's a lot to do here, a lot to build. Seeing V0 come up with the baseline so quickly was amazing.

Nancy Wang: Going back to the observability comment you made, and potentially hinting at future product features - for engineering teams, debugging this in real time, what tool should they use? How should they do that as V0 is generating these new baselines?

Tom Occhino: Yeah. I mean, certainly once you're in production, there's just a bunch of best-in-class observability tools that allow you to see what's happening in production. I'm using V0 here to observe what V0 is doing, but yeah - Vercel has a great observability suite if you're deploying to Vercel. But I think the beauty of this is you can tell the agents what observability tools you want to use and they can build those integrations for you. If you're not familiar with any, you can have it recommend one. You can ask the agents to teach you how to use it. Observability and telemetry are going to be increasingly important over time. But yeah, you can kind of cook up whatever you need - ask questions and explore it.

Dev Tagare: Wow, that's amazing. The other thing I really love - and this is probably cutting over into Nancy's world - is the human-in-the-loop feedback, because now you can interject, you can guide it however you want. It waits for "Is this the right thing I've done?" once in a while. I really like it because it's the only realistic path to abuse mitigation at this point. Otherwise you could have bots just go in and keep doing their thing without any human oversight whatsoever. So it's just the right amount of human-in-the-loop feedback.

Tom Occhino: Yeah. And I think we want human in the loop for a long time. I mean, there are certain things that I'll be developing where I'm just like, no, this is my own thing, it's personal, it's small. But I think for any production system, it's going to be really, really critical that everything is not an afterthought but really part of the core flow - "Where do I decide?" And the best thing that the agents can do is provide us with as much context as possible to be able to make the right decisions.

I've got some scripts to run here to generate our database. I'm going to just run that and let it keep going. I can't believe we're done already - we used V0 Max here, so this was a big one. But let's - you guys okay with taking a look at our app?

Dev Tagare: Let's do it.

Tom Occhino: Cool. So this is our financial portfolio tracker. I don't know what features we have, but let's look. We have a watchlist. I can add something to the watchlist. Do I already have Apple? Okay. Company name? It should just probably fetch that - I would add that enhancement. Oh, and add it. Perfect. Critically, this is all stored in a database. We're going to take a look at that in a second.

Here we're doing a very live demo, but I want to show you how this is a real production app that can be deployed. I'm going to click "manage" on our database and click "configure" - it'll take me into the Vercel dashboard. What's really cool is, let's say I'm building an internal app. When I build this, the biggest thing that needs to happen is visibility and access to this thing is really, really important. So if I'm building an internal tool for my company team, I need to make sure that only my colleagues can see it. And so this kind of locks it down by default.

What should we look at in our app? Maybe we try adding a widget or two to this.

Dev Tagare: Yeah. Let's say we do like a trend line of analysts' forecast to stock price for the last six months.

Tom Occhino: Actually, that reminds me -

Nancy Wang: Dev, you used to work at Robinhood, right?

Dev Tagare: I worked at Robinhood and at East Laurel. This gives me a lot of memories of using Bloomberg, where - it was a skill, you know, in 2010 precisely. It was a skill to operate the shortkeys on a Bloomberg terminal to basically build a view like this. And now I'm like, I can give a handful of prompts and just render it. So thank God I went deeper into software engineering.

Tom Occhino: But let's see if I can get this right. So do you want to say it again?

Dev Tagare: Let's do a widget that plots the trend line of analyst consensus stock prices versus actuals for the last six months. From whatever list you have of stocks.

Tom Occhino: Let's see. Yeah, I think it's going to work. I think you have all the information for this in your database. Yeah. I'm surprised it's live-updating in real time - I don't think I asked for that. It's working. We can still play with it.

One of the things I also want to show is the design mode, which is really cool - you can go in and make changes to the design of the whole thing in a very visual way. I love the interactive thinking mode, but it's not overly verbose because the moment you're broadcasting all the thoughts the model has back to the user, it's way too much.

Dev Tagare: Yes, it's doing a ton behind the scenes. I think you could introduce a verbose mode for anybody who's really curious about what's happening. But there's a lot happening behind the scenes that we just see as "adding analysts consensus chart." A technique I've seen is instead of the steps you see there, you click through that and then you enter the verbose mode.

Tom Occhino: Yeah, absolutely. And you know, when it's doing these reads and telling you what it's reading, that's not as interesting as what's happening behind the scenes here.

Dev Tagare: Yeah, I mean, this is incredible. Quite a good UI too.

Tom Occhino: So this is all things. Like, pick the holding and then -

Dev Tagare: Got it. Those are either missed opportunities or dodged bullets, however you look at it.

Tom Occhino: Yes, depending on the direction. This is actually a really cool chart, and it seems like it understood the assignment too - because this is what this analysis looks like. If you're growing with the trend you want the two lines to be somewhat similar.

Dev Tagare: Amazing. Wow. This is super cool. I mean, I guess it puts more emphasis on live demos, right? And going back to the verbose mode, I think it could be helpful for developers to understand why a certain UI path was taken.

Tom Occhino: Yeah. Oh, yeah - that's for sure. Absolutely. One of my favorite things to do here is to ask questions about why it did something a certain way. I have this example - I'll stop sharing my screen now. I was building this very, very simple app. It's just for counting things. You type in a thing and it creates a button, and every time you hit the button it counts up. I needed this to count light switches - but I've since used it when people are giving presentations to count repetitive words. I'll type the word and say, you said it this many times.

Very simple app. I'm the only one who uses it. But what was really interesting about this app is on mobile web at the time, you can't do haptic feedback, and I needed some way to know that the click registered without me looking at it. So I wanted to add sound. I asked V0 to add sound every time a button was clicked - I could have looked this up, I've done this before - but it was so cool because it did it in a way that taught me how to synthesize sounds in the browser by creating a synthesizer. So I'm reading through the code it wrote and I was like, oh my gosh, this is so great. So I copy-pasted that code into another existing app because I just wanted it to make sounds when the buttons were clicked. And I learned how to do this whole thing. So asking it how it did something, why it chose to do something a certain way - I think this is a really powerful new way of doing things.

9. Prediction Round: What Gets Replaced in 18 Months?

Nancy Wang: You know, we can certainly go for much, much longer. But I did want to end this episode on a prediction question.

Tom Occhino: Of course.

Nancy Wang: And Dev has a fun question for you.

Dev Tagare: So given just the speed of iteration, what part of the AI stack do you think is going to become obsolete in the next 18 months? And - get replaced with Vercel is really the subtext.

Tom Occhino: I think there's a ton of things that people don't genuinely enjoy about their jobs, and I hope that Vercel replaces all of the things that people hate about their jobs. I mean, I used to hate diagnosing an incident - trying to figure out root cause. Something happens, there's a spike in errors. So well, what did we build? We built agentic investigations of anomalies. So: any time a chart spikes in either direction and something looks off, it triggers an alert and we tell you about that alert in Slack. I get an alert that says, hey, there's a rise in 404 errors here, there's a ton of blocked requests coming over here. There's a spike and immediately an agent goes and investigates it and tells me exactly what happened - way better than I could have done. Our CTO used to really enjoy doing these investigations but just didn't have time. I used to hate doing these investigations.

Anything that is the worst part of your job - I think we want software to replace that, so we get to focus on the fun part. I like creating things and building things. Code review is another one - I don't mind doing code review, but I'm not that good at it. So this is another place where anything humans aren't as good at as agents can be, that's ripe for us to move aside and focus on the things we're good at and enjoy doing rather than these other things. But I'm curious about your predictions - what do you think is going to be replaced?

Nancy Wang: Well, let's say replaced by Vercel - I mean, just seeing the example you pulled up with V0. Prototyping tools out there, I'd be concerned. And look, if you build in a rich enough ecosystem of telemetry, observability, debugging - you already have the sandbox - that gets pretty fleshed out. So that's what I think. What about you, Dev?

Dev Tagare: Yeah, I think I have a maybe slightly spicy take on this - the inversion of what Tom said. I think the way developers will access agents will be through interfaces and not through agents themselves. So the imperative style of programming that we're using today to build agents will actually flip. And the interfaces would be something like Vercel, where I'm building the application directly and not bothering with what agent to build. That piece will just be done by a combination of the SDKs we have today and the models getting more and more intelligent over time. So my hot take is an inversion in the model that we have with agentic AI tech today.

10. Lightning Round: What Would You Build?

Nancy Wang: All right. So one last question, Tom. You're a builder. I've always had this itch that it's probably an itch that you haven't scratched yet. So if you weren't in your current role and you had a free month, what would you build just for fun?

Tom Occhino: You know, I actually started building it. My wife and I used to use a text document to track all of our favorite things - all of the best things we have. It started out with the best foods, because one time I had a dessert that was really good and I had to write it down. So I started building an app - actually, I did this over the holidays. It was me plus V0 plus my buddy Claude Opus. And I could turn this into a whole thing. It's completely overengineered now - it's got the most sophisticated auth on the planet for something that only two people will ever log into. It's got a really nice database schema that doesn't matter. But I like building software that I use. I don't necessarily care about distribution. I'm not trying to build something that works for everyone. I really love building software that becomes a part of how I operate and how I live my life. I've built a few apps like that where it's like personal software.

This app called "The Best" - if I spent a month on this, it would end up being like the most amazing app of all time. But even in just a couple of hours a day for a few days, I built a whole app and I'm using it. So yeah - personal software. Sort of like moving from personalized apps to personal apps.

Dev Tagare: That's right. That's an interesting journey.

Nancy Wang: Well, Tom, again, I wish we had many, many more hours - because I do actually want to dig into what auth methods you built. But thanks so much for coming into our studio and spending time with myself and Dev. Big fan of what you do, what Vercel does for builders and aspiring builders everywhere.

Tom Occhino: Thank you. Thank you so much for having me.

Zero-Shot Learning is presented by 1Password.