EPISODE 893

[INTRODUCTION]

[0:00:00.3] JM: A software engineer often works as a contractor for some duration of their career. A contractor earns a fixed hourly salary for a defined period of weeks, months or years. Contract work can be more flexible than full-time work and it often pays more than full-time software engineering, because contract jobs can end at any time and they do not have the added employee benefits, such as health insurance and stock options.

Online contracting platforms, such as Upwork and Fiverr have expanded the number of software contracting engagements which take place. Developers on Upwork and Fiverr have a wide range of skills and experience levels. The clients who come on Upwork and Fiverr are looking for developers and they're sometimes unsophisticated at managing the software projects that they're hiring for.

In some cases, software developers even get defrauded on these hiring platforms and they get tricked into doing free work. This is not really the fault of the platforms, it's just an inevitable side effect of any marketplace platform. You get chargebacks, you get some amount of fraud, you get some amount of problematic behavior. The question is how to improve the economics, such that there is less fraudulent behavior and less dissatisfied participation on the platform?

Moonlight is a contracting platform for software engineers. Today's guests, Emma Lawson and Philip Thomas are the founders of Moonlight and they join the show to explain why they started the company and the gaps that exist in the world of software contracting. Moonlight's model is different than most other contracting platforms, in that Moonlight requires clients to pay a $300 subscription fee to recruit engineers on the platform.

This $300 subscription price lowers the rate at which clients can take advantage of software engineers. The dynamic of the subscription price has downstream impacts, like causing the software engineers to take their work more seriously and act more professionally.

Full disclosure, I am an investor in Moonlight. I've also been a paying client of the service. If you want to see the results firsthand, you can look at our Software Daily Android app, which was built by a Mostafa Gazar. He's a talented Android engineer that I met on Moonlight. I am a fan of the service. I'm going to be continuing to use it going forward. Again full disclosure, I'm a small investor in the company. I hope you like this episode.

[SPONSOR MESSAGE]

[0:02:42.1] JM: Monday.com is a team management platform that brings all of your work, external tools and communications into one place, making cross-team collaboration easy. You can try monday.com and get a 14-day trial by going to monday.com/sedaily. If you decide to become a customer, you will get 10% off by using coupon code SEDAILY.

What I love most about monday.com is how fast it is. Many project management tools are hard to use, because they take so long to respond. When you're engaging with project management and communication software, you need it to be fast, you need it to be responsive and you need the UI to be intuitive. Monday.com has a modern interface that's beautiful to look at. There are lots of ways to use Monday, but it doesn't feel overly opinionated. It's flexible, can adapt to whatever application you need, dashboards, communication, Kanban boards, issue tracking.

If you're ready to change the way that you work online, give monday.com a try by going to monday.com/sedaily and get a free 14-day trial. You will also get 10% off if you use the discount code SEDAILY. Monday.com received a Webby award for productivity app of the year and that's because many teams have used monday.com to become productive; companies like WeWork and Phillips and wix.com.

Try out monday.com today by going to monday.com/sedaily. Thank you to monday.com for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:04:34.4] JM: Emma and Philip, you are working on Moonlight. Welcome to Software Engineering Daily.

[0:04:39.7] PT: Thanks, Jeff. Excited to be on.

[0:04:41.4] EL: Thank you for having us.

[0:04:43.1] JM: The subject of this discussion is going to be contracting and contracting platforms and specifically what you're building at Moonlight. Can you give me a brief history of the platforms that have been used to allow software engineers to do contracting through online marketplaces?

[0:05:04.6] PT: Yes. Looking at the history of contracting, what we observed in the market was that a lot of existing websites were really optimizing for decreasing cost for the clients. As software professionals, we realized that that wasn't a great way to build software. That the cheapest bid is not always the right way to go. That was one of the early ways we were interested in starting Moonlight was I previously had a software startup that needed to hire some really experienced people going into it. I wanted people that were smarter than me in areas like DevOps and front-end and back-end, which isn't a very high bar to pass, but I was instead going on other websites and just seeing people really bidding for the lowest price possible. That's not what I wanted to optimize for.

I really wanted to meet qualified professionals who were passionate about what they were doing and I compensate them fairly. I think at the other end of the market, you have full-time placement websites that charge enormous contingency fees, of tens of thousands of dollars, and really understand the value that a good developer can bring. Particularly from the contractor market, there weren't really sites out there that respect the individual makers and compensate them at a level that would be equal with full-time work on other websites.

[0:06:30.5] JM: This is very consistent with my personal experience in hiring contractors, because I've used a lot of these contracting platforms, because I just think they're fantastic. They actually do align incentives in a great way, despite the bad reputation that contracting, or “outsourcing” got in the 90s, or has gotten overtime.

I think that the persistent problems with outsourcing have largely been due to the platforms, rather than the actual fundamental incentive alignment that you can get out of a contractor-employer relationship.

[0:07:09.0] PT: Yeah. One of the ways that these marketplaces often operate is having to preserve their position as a middle person. Needing to limit communication between the two sides of the marketplace, in order for them to make money basically; we really think that that's not how the world should work. We realized after a running Moonlight, that letting people make connections on the website and work together directly is really the most powerful way to build companies.

[0:07:42.5] JM: What have been the other issues of these contracting platforms in the past? We don't need to name names, but just to refresh the two kinds of ends of the marketplace, I think you laid it out accurately, there are these total open marketplaces. Actually, I'll go ahead and name names. You got Upwork and Fiverr. I love Upwork and Fiverr. I've talked about how much I love these platforms many times on the podcast, but they're a one-size-fits-all solution for contracting all kinds of things.

Software engineers are a different breed. Then you have these very elite contracting platforms as well, where you have a very expensive engagement, but you're guaranteed a high-talent developer. Those are the market dynamics. Are there other issues, like features, or just things that are endemic in these platforms that you saw as perhaps UX issues?

[0:08:36.9] EL: I think a major thing with the low-end of the market is that it becomes a race to the bottom for contractors. For example, when we were starting to look for contract work after we left the Bay Area, it was really hard to find the work we were looking for, because a lot of the work was getting outsourced to companies who would have a frontman, who was speaking really good English and selling this vision, but then outsourcing that work to people further down the line who were charging maybe somewhere between $10 and $20 per hour. It really cuts out the high-end of the market.

Then on the other end, you have people controlling the relationship. At the higher end, there's a product manager in the middle, who is finding a developer and then controlling that whole relationship. You really don't get to work directly with someone on the product that you care so much about.

[0:09:31.2] PT: That's where Moonlight has really focused on matching. At the core of it, we are not going in and running every developer through a coding test. That's one-dimensional. Our point of view is that Moonlight should act like a dating site almost under the hood, where if the developer is really good at Shopify and a client is looking for a Shopify developer, it's our job to try to match those people, recognizing that software is moved towards increasing levels of specialization in recent years. That as work becomes more distributed and remote, if you have every developer in the world able to apply for every job in the world, then under the hood, you really need to be focusing a lot on relevancy and matching and really – or connecting these disconnected people in an effective way.

[0:10:23.9] JM: Describe how your company makes money in this situation, because it's a very different set of constraints that you're putting on the contractor and employer relationship. What's the economic model for Moonlight itself?

[0:10:41.8] EL: Yeah. At the beginning of Moonlight, we were doing the same thing as a lot of the other platforms we're doing, which was taking part of the hourly rate. That's why in the beginning, we were pushing people to have higher hourly rates, not help the quality of the developers on the platform go up.

About six months ago, we switched over to a subscription model. Now companies join Moonlight as either a monthly or annual subscriber. We don't really care about as much about how they work together, or how much they're exchanging through the platform, because we make money on people just engaging inside of the community. It's really increased the quality of engagement. Once somebody is subscribed as a company, they come back to Moonlight time and time again as the first place to find new technical talent, rather than a last resort, because their personal network didn't work. It really aligns incentives, so that we don't have to do anything to control how people are working together. They're really just doing the best thing for their company.

[0:11:47.9] PT: I think that's a lesson for anybody working on a startup listening to this podcast, is think about whether your pricing encourages usage. If we were charging a flat rate per hire, which we've done in the past, we would have happy clients come in, hire somebody, be really satisfied with the service and then we'd ask them if they wanted to keep using Moonlight for the next hire. Then say, “Well, maybe. I'm going to look at all the options again, because I have to pay you again for every hire.” Whereas with subscription pricing, we are not limiting the number of hires that somebody can make. It's getting clients to come in, make one hire and then realize that they can keep hiring for free after that basically. That pricing really encourages people to use the platform and doesn't make it so that we're having to continually fight for new business from existing clients.

[0:12:37.0] JM: One thing I like about the story of how Moonlight has developed is that it's quite a good case study in startup pragmatism. You do hear these stories and these tactics about building an MVP without much code being written, like putting together a Google Doc, or putting up a Squarespace site or something like that. Then you read the stories of people actually putting these things together, it's amazing. Moonlight is one of those stories. You have a great write-up on Indie Hackers that I've read twice now, just about how you can build stuff without programming.

You can stitch together really sophisticated platforms. I mean, I find this movement of the low-code/API/higher level platforms/WYSIWYG drop-in things, I find this really inspiring because it lowers the bar for people to start businesses and to build online platforms and to validate their ideas in a very serious way. Moonlight is a great case study in that. Then you of course, you’re capable product designers and software engineers, so you you're able to actually swap out the rigid, higher-level Squarespace components for eventually your own custom built platform.

I think that's a really interesting story and I think it's a underexplored pattern for validating startups and then getting startups to market. I think, plenty of people have done it. Just for the sake of exposing people to another case study, can you describe the first version of Moonlight and how you stitched it together without programming?

[0:14:23.8] EL: Yeah. This is a topic that both of us are really passionate about, just because it's the reason that Moonlight has succeeded and we've been able to quickly iterate and evolve to what it is today. The beginning of Moonlight was really us manually doing a ton of admin work. None of it was automated. We had a Squarespace site, we had a type form, we e-mailed a bunch of developers to validate the idea. After we had about a thousand signups within a couple of weeks, we decided to move forward with the idea.

We got clients on one side to fill out a type form, where they asked us for a certain type of developer based on a task. Then we would take the type form of they came through on the developer side. If they had said they were an expert in React, we would e-mail that mailing list of all the React developers, let's say it's a 100 people. Then we would send the job post to them. They would respond and we would connect them directly through e-mail, and then we just send them an invoice using a product through Stripe. Then sometimes we would PayPal out to the developers in the beginning.

We slowly automated this overtime after we had done it hundreds of times and really understood the admin side of the process. It wasn't until we were about a year in that the product was really more automated in the sense that we didn't have to be manually doing matching, or invoicing, or anything like that.

[0:15:52.8] PT: I think one of the takeaways with this is just really falling in love with the problem, rather than a particular solution. We have been working on Moonlight for over two years at this point. The first two years were definitely not glamorous. It was a lot of iterating, a lot of talking to users. The thing that no-code prototypes let you really do is experiment really quickly without feeling you have this sunk cost of a ton of code, or a big application. I think that's one of the things that a lot of people don't talk about is just how long it takes to get to the overnight success.

I've been watching Superhuman this year. They're an amazing e-mail client. They're really blowing up the news in the past six to 12 months. According to Crunchbase, they started nearly five years ago in 2014, and it took them probably years of iteration and experimentation to get to where they are today. Now it's finally starting to take off.

With Moonlight, when we started it two years ago, it wasn't an instant success. It took a lot of iteration and research and no code was a great way to focus on that quick iteration, but it really can take a long time for a business to take off.

[0:17:05.2] JM: Describe the first version of the product when you started coding. When you decided to start building an engineering solution around this, did you have it be half a software platform, half low-code platform, or did you just do an all-in, switch over the entire thing to a new Node.js application or something?

[0:17:32.1] EL: Yeah, so we have always done things very slowly and iteratively. I think that was because we were a bootstrap company for so long and it was just two of us working on it. When we started, we had profiles. The only thing that was part of the product was a developer could sign up, put in their information and create a profile. That's still the same profile we have today. There's a lot more we can do with that. It let us just experiment with what would it look like if we just started sending e-mails to clients with developer profiles and then they could hire them from that profile.

Then we slowly added on invoicing and matching and a fully automated newsletter, all of these pieces that let people do a full contractor process. Then now that we're hiring on a more fully built out product team, we're going to be able to move a lot faster with moving towards more full-time remote hires and letting people collaborate together. It took us a year just experimenting with those profiles and a little bit of invoicing to get to the point where we knew exactly what it needed to be.

[0:18:40.3] PT: We continue to use no code solutions today. The application has never had zero no code solutions. It's just had progressively more in-house software, but we still send people surveys through Google Forms and through type forms. We use no code systems to prototype the next things we're going to build. Some of the things you'll see in Moonlight in the coming year came out of Google Forms that we're using right now.

[SPONSOR MESSAGE]

[0:19:15.7] JM: GitLab is the open source platform for DevOps. The success of GitLab has accelerated the adoption of DevOps across the enterprise market. GitLab Commit is the official conference for GitLab and it's coming to Brooklyn, New York, September 17th, 2019. GitLab Commit features discussions of technology, lessons learned and a close look at the DevOps lifecycle, including Kubernetes, CICD, DataOps, security and remote culture.

GitLab Commit is September 17th in Brooklyn. You can save $99 on your pass by going to softwareengineeringdaily.com/commit and entering the code COMMIT99 to save $99 on a conference pass, if you register by August 15th at 11:59 p.m. Pacific time.

GitLab is an exciting ecosystem of products and it's accelerating the world of DevOps. If you can make it to Brooklyn on September 17th, check out GitLab Commit and go to softwareengineeringdaily.com/commit to get that additional $99 off by entering the code COMMIT99.

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

[INTERVIEW CONTINUED]

[0:20:39.0] JM: This is slightly deviation from Moonlight itself, but when I look at things like Airtable for example, I look at Airtable I'm like, there's opportunities for software engineering organizations to have this software be middleware within the company. There's opportunities to replace parts of a API surface with Airtable. Why make a call to a node.js application when you can make a call to some Airtable-based function as a service?

I haven't engaged with Airtable or some of these other similar platforms enough to understand where they're at in terms of usability today. I wonder if you have any perspective on how this might change software engineering.

[0:21:36.0] PT: Yeah. I think that one of the other factors that comes into this is people expecting a higher level of polish on applications. I guess, that's a part of the Superhuman letter that David Ulevitch wrote was talking about the prosumerization of enterprise. Just that consumers have this expectation of really high levels of polish for applications. I mean, the Facebook app, they'd probably use daily, has thousands of designers working on it full-time. Enterprise is starting to have some of that expectation of additional levels of polish trickle over.

I think that is one of the things that's starting to drive outsourcing of software tools a little bit more, is just that you could build a form in-house. If you're trying to manage things like accessibility and mobile functionality and things like that, there just start to be a lot of overhead with that. I think that this whole area is really interesting. It's almost like microservices playing out with third-party applications.

Another tool in the space that I think is super promising is Retool. They are a YC company building a way to handle internal tools. Every application behind it has some support dashboard basically, where you can deactivate user accounts and approve things and things like that. They're just trying to make that generic internal application software that you can buy. From what I understand, there's a lot of big companies that instead of building and maintaining internal tools, are just starting to build on top of Retool. I think that's really fascinating. We're even seeing things like newsletters and whole applications being built with only Retool.

[0:23:18.7] JM: I've heard the same thing. One area I find interesting is the fact that a lot of this stuff is not open source. Developers are – I think there was perhaps a strong bias for open source for a little bit, like after the hangover from the post-Microsoft stymieing of open innovation. Then we went so far in the direction of everything being open that we started to realize, there's actually incentive alignment when you hire a company to fulfill a task for you. AWS was this at the most atomic level.

Then over time, it's just moved higher and higher up the stack where you started to say, "Look, it's actually not that bad to pay MailChimp 50 bucks to manage my newsletter every month, or it's not that bad of a deal to pay Airtable a 150 bucks a month to manage my spreadsheet-based microservices architecture," because these things are – they're affordable. I mean, if you're if you're a high-margin software business, or that's where you're cruising towards, you can afford these monthly costs.

[0:24:30.2] PT: I think it also lets every company utilize technology without having to be a technology company with a software engineering staff. Airtable lets consulting companies keep track of their contacts, Squarespace lets small businesses have websites. I think that trend is really important, because these tools are becoming so easy to use that you don't need to be a technology company, or a technology-native company, or even a venture scale startup in order to leverage some of these really cool technologies.

[0:25:03.2] JM: One advanced high-level tool that I think Moonlight uses in an interesting way, a way that I certainly had not seen before was how you use Slack. I think the way that most people view Slack is like, "Okay. Yeah. This is my team chatroom for my team. I can switch to channels, here's the DevOps channel, here's the software architecture channel." For you, you have it more as this permission-based room system. It's very sophisticated to you. Can you just explain how you use Slack? Then also, just since we haven't really gone deep into Moonlight, explain how that fits into the current product of Moonlight.

[0:25:49.8] EL: Yeah. I think, Slack is an example of how we just wanted to test something as an MVP. We used Slack as we know everybody is on Slack and people like using it and it's a really easy way to collaborate on remote work. Instead of building a chat tool from scratch, because a lot of other tools out there that you can build into your application are very expensive. We just decided to lean pretty heavily on the Slack API.

Our app is pulling the messages that come through Slack and then pulling them into our own application. The way most users on our platform are interacting with other people in the community is through Slack itself. Then if they want to, they can use the notifications in our app. I think this was a really great way to test out the community and see people engaging and keep them inside of the marketplace, instead of going around us through e-mail or whatever other platform. It's an example of something we probably won't lean on for very much longer, because we want to be able to do more custom things with it. We never could have seen the traction that we saw without having native applications, without being able to use Slack as that testing ground.

A specific example of that is a client joins. They want to talk to a developer about a job. They join the membership and then we let them talk in a private channel that says like, "I'm a software engineer interview," and then that's how they can collaborate. It's on their device. It's on their computer. They're getting notifications. It wasn't something that we had to completely build from scratch to test out this idea.

[0:27:31.5] PT: In the technical side, what we're trying to do is basically build a lot of box and toolings on Slack, just through Webhooks. That lets us maintain contacts from the application, but boot up these private channels for Slack. Basically tie every single Slack user to a user account in Moonlight. A lot of that was inspired by Nomad List. Pieter Levels did some really cool tooling on top of his Slack community, so he could do moderation and ban people and things like that. Slack has a really cool API that you can use with their free plan to do a lot of different stuff, which is really cool.

[0:28:10.3] JM: This result, you were able to build all that on the free plan? Slack's free plan?

[0:28:15.5] PT: Yeah. I mean, we have well over – I'm not sure the exact number, but well over a thousand people in our Slack community. At 10 bucks a head per month, we definitely don't want to be using the premium plan for that. Or we can't logistically make sense paying for that.

[0:28:31.4] JM: Okay. Let's get more into the engineering. Can you describe the software architecture of Moonlight as it stands today? Perhaps if you can touch a little bit more on the process of refactoring the application from its low-code form to the current engineering solution that you have today.

[0:28:53.9] PT: Yeah. Moonlight has been the same codebase since it started. The back-end is written in Go. We leveraged Kubernetes really heavily, using a MySQL database, some Redis caching. Then front-end, it's a big Vue.js single page application. The repo basically started when we got rid of Squarespace and started tell posting the website. It's just slowly grown in complexity.

We use a single codebase, a mono repo for everything, which is really nice because we use GRPC as our API. We're able to simultaneously make changes on the front-end and the back-end. That's been really helpful for being able to refactor things. Most of our code is in single monolithic application. The nice thing about Kubernetes is that it's let us add additional services into our hosting that were not necessarily leveraging for the core applications, like using Google Chrome in a Node.js Docker container to render invoices as a PDF, or something like that.

[0:30:02.7] JM: Oh, wow. All right. Well, you condensed a lot of interesting points into a single answer. Wow, so much to go into. Go on the back-end and Vue.js on the front-end. What's with those preferences?

[0:30:19.9] PT: I've previously used Go at multiple companies. I really like having strong type things within our stack. MySQL, Go, GRPC, it just reduces a lot of runtime errors and complexity. The nice thing about Go is that more or less, if it compiles, you prevent a lot of runtime problems in general. It's not quite as safe to something like Rust. Compared to problems I've had in the past of having to SSH into a running container, to debug, dependency problems or something like that, it's been really nice.

Then on the front-end, we used Vue.js, basically because it was the most modern technology I knew. Did a really great Udemy course on Vue.js and realized that it had great documentation in the core libraries for a lot of the different tools you would need from a state store to a router. It was definitely something that you could build a full application in. After a while, we have just had to stick with it, because we have what is now a fairly large application in Vue.

[0:31:30.1] JM: Do you have any regrets? Our website softwaredaily.com, so we have Software Engineering Daily, which is a WordPress. We're going through this up the exact same thing, where we're over time migrating to our own platform and Software Daily is this thing that basically scrapes our WordPress and rebuilds our site as a Node.js Vue application. I've programmed a little bit in Vue and I love it. It's great. The network effect of React is so strong. There's so many things in React that you can just take off the shelf. I'm sure that's evident in Vue as well, but it's a power-law thing, right? React is way ahead of Vue, in terms of things you can take off the shelf. Do you have any buyer's remorse there?

[0:32:19.7] PT: Well, looking at the Moonlight network in general, we see a lot of people using React, so that they can leverage React Native. I think that's a huge value add for React. We have a lot of clients that run full stack JavaScript code bases, including web, Native Android through React Native, Native iOS through React Native, Node.js in the back-ends and that's something like Mongo for their database. I think that's super powerful. Definitely an amazing way to build an application.

I also wonder what's going to happen in the next five years. I think there's going to be a lot more push towards progressive web apps. I don't know if that's necessarily the solution for everything. It's definitely not necessarily going to be the same level of polish as full native apps. I'm sure Emma can comment more on that.

I think that we're going to see a lot more democratization of JavaScript apps on the front-end just through PWAs, because you get a lot of those value adds like a native app without having to go through such high overhead of an app store and things like that.

[0:33:23.9] EL: Yeah. Just to add to the native capabilities, I think from a business effective I see a huge reason to use React, so that you can have native apps in a pretty easy way. From a consumer and design perspective, I don't think React apps are super responsive, or easy to use. As companies like Apple create new components, it's hard to keep up with it in the React framework, I think.

I could see an argument for us just deciding to build native applications on iOS and for Android, instead of trying to leverage one of the web technologies, especially if messaging becomes a huge part of our application. I think using the native components is much more powerful, if you can afford it.

[0:34:11.0] PT: Also comparing Vue and React just on a web stack, I think Vue tends to be way more Omakase a style, like Ruby on Rails, where it has all the basics you need included in that standard library. There is a big benefit to that of not having to at the beginning of a codebase, go through and take your state store, pick your router and pick a bunch of different frameworks. You can just get started with Vue. There are huge benefits to React from that ecosystem, such as React Native clearly.

[0:34:40.8] JM: Just to go a little bit deeper there, the React ecosystem, what I hear about it and I have not been a React programmer is that you can take a lot of components off the shelf. If you need an audio player, for example, there's a great audio player that you can just import as a UI component, it plugs very nicely into the back-end. Is there truth to that and does that contrasts with the Vue community?

[0:35:06.0] PT: I'm not the best expert on this, but what I always think about is that it's possible that Vue.js has more daily active users than React, just because of how much it started to take off in China. I mean, some really large companies like Baidu are leveraging Vue.js really extensively.

I think that the Vue.js community has a lot of active participants and a lot of interesting things going on. Yeah, React definitely has a lot of components. If you're looking for very particular things, they definitely have it. The nice thing about component-based architecture is that it's pretty straightforward to build some of your own libraries and we have done that a lot; things like Google sign-in, or different integrations with third-party JavaScript libraries and things like that. It's not horrible to build as your own components internally.

[0:35:56.9] EL: Yeah. To be clear, we're not really stuck to Vue. If there was a really good reason for us to switch to something else like React, we would definitely do it. Neither of us are front-end engineer specialists. We just hired somebody who is a really good unicorn mix of a designer and a very technical front-end engineer. We're hoping that he's going to have a lot more clear ideas of what we should be doing moving forward.

I think the thing that we really want to stick to is language that lets us build with frameworks and components, so that we can just make sure there's a really consistent language throughout the whole app. I could see us being convinced to switch to React at some point.

[0:36:39.8] PT: Coming back to the Moonlight community in general, we see people asking us all the time like, “Should I use React, or should I use Vue? Should I use Rails, or should I use Django?” Our general advice is typically that, what you know, or what your team knows is the best thing to go with in general, because you really understand the capabilities much more in something that you're specialized in, than necessarily trying to shop around for the perfect framework. Most teams really don't have the opportunity to teach everybody internally a brand new system. Whenever we hear about clients or startups that want to do a complete re-architecture, it's really not always the best decision and can typically take a lot of time.

I think for pre-product market fit companies in particular, it is more about what you know internally, than what is necessarily the best long-term choice. We still see applications that are being built in PHP and doing amazingly. I don't think that anyone would know on the front-end that it's a PHP application. At some point, you just have to get product shipped. Especially for pre-product market fit companies, speed of delivery is way more important than those long-term architecture decisions.

[0:37:56.3] JM: 100%. Especially, because V8 is so good. The JVM is so good. All these underlying – the browser is so good these days. Everything works very quickly. If you have an application bottleneck, very often it's just because your code is written in a way that's not performing. You don't need to change your framework. You just need to refactor the for loop or something.

I want to talk a little bit more about the marketplace and platform and the competitive landscape. Actually, I guess we should talk about the product a little bit more, because people probably – most the people listening probably have not seen the product. It walks this interesting line between marketplace and collaboration tool. When I look at the product and I interact with it and I'm a customer of it. I've hired the Android engineer Mostafa for Software Engineering Daily from Moonlight and the experience has been wonderful. In my interaction with the platform, I felt there was a tension between marketplace and communication tool. I'm just wondering if that has come out as attention in terms of what features you've built.

[0:39:12.9] PT: Yeah, that's definitely a great observation and a great question. Moonlight is a professional community for software developers. That's how we see it. We're really primarily working with software developers. Our long-term goals are focused a lot on objectives beyond just short term contract work. We're doing a lot more work with full time hiring, helping companies build full time remote teams in particular. We're also working on different initiatives to go beyond just jobs to service remote workers.

To answer your question, I think if you're looking at any marketplace, the two main places where they create the most value are going to be in sourcing and in payments. Those are two places where Moonlight has really focused. Sourcing, meaning helping you hire somebody, or meet that person that you want to hire. The other really great value add is in the payment side, which is making it easy to send payments to people. That's really the two main areas that Moonlight has focused on up until now, is being a sourcing product so that you can get started on the project and helping to make the payments much easier as you continue to work with somebody.

[0:40:21.7] EL: As far as your question about messaging and communication and collaboration, one thing that we learned very early on is that if it's not easy for people to communicate and it's not clean enough of an experience for somebody to get from point A to point B, they won't stick around. They want to be here because we have amazing talent. In order for us to keep them and retain them as an ongoing user, which is our goal, it really needs to be an entire experience that allows them to do more than just find a person.

The matchmaking is the important thing that we focus on now. To keep someone engaged and using Moonlight for more than just that first interaction, it really has to be a much more complex ecosystem. That's where some of those newer features have come in, that we think are going to be really important for a long-term growth.

[0:41:18.1] PT: A lot of the features in the application really are built around optimizing productivity that you can just get to work. When you hire somebody through Moonlight, our Terms of Service provide a standard contractor agreement, including an assignment agreement and things like that. We already have all the payment information set up, so you don't have to necessarily add someone to your payroll system. We have the Slack channel, so you have this layer – this shared Slack channel, so that you're able to just start communicating. You don't have to go through the step of adding a contractor to your internal Slack channel. Particularly for short engagements, we're trying to really make it easy to just start working as quickly as possible using the tools that you already know.

[SPONSOR MESSAGE]

[0:42:05.5] JM: Over the last few months, I've started hearing about Retool. Every business needs internal tools, but if we're being honest, I don't know of many engineers who really enjoy building internal tools. It can be hard to get engineering resources to build back-office applications. It's definitely hard to get engineers excited about maintaining those back-office applications.

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

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

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

[INTERVIEW CONTINUED]

[0:43:45.2] JM: What is beautiful about the incentive structure is that it allows for my favorite way of trying out a developer, which is contract-to-hire. This is a strategy that I find – it's perplexing to me how underutilized this is, or maybe it's more utilized than I realize perhaps. Just the strategy of bringing on a contractor and working with them and then deciding whether or not to hire them full-time, that's so much better than the interview process. I find it's so much less of a time sink.

You think about all the amount of money that goes into hiring processes, where somebody comes in for a full round of interviews and it sucks out basically 10 engineering hours from all the different people who have to take time out of their day to interact with this engineer and this this candidate and then decide whether or not to hire them and so on. Just bring them on as a contractor. Give them some restricted privileges to your product, to your infrastructure and then see what it's like to work with them.

You can enable people to do that more as a service with Moonlight. Can you give me your reflections on the contract-to-hire idea and whether you think that's under-utilized and how predominant the usage of contract-to-hire is on Moonlight.

[0:45:18.7] EL: Yeah. It's definitely the main process that we try to get companies to interact with developers through the marketplace with. Philip and I have both gone through a lot of interview processes as developers, so going through a white-boarding test and trying to prove your worth in a very theoretical way. That's where we both didn't really believe in that process, as much –

[0:45:43.7] JM: It’s madness. It is complete madness.

[0:45:46.3] EL: Yes. Especially for a small team, it takes much time. It's really hard to understand from both sides. Just respecting someone by paying them for their time is the number one thing. Then also jumping in and working on a real-world situation inside of your application, just tells you so much about a person that you never could have learned outside of that. We're even using that for our internal hiring process right now. There were a few candidates who we were going to move forward with and made decisions based on this contract process, or we saw something that was amazing, like they overwrite their documentation in an amazing way and we loved that and we never could have learned that from just doing a test project or a white-boarding session with them.

With remote work especially, it's so important that someone has just as much emotional intelligence as technical intelligence. Making sure that they communicate in the right way and they can do written communication and can communicate over a video call. All of these things are not technical, but they really matter when you're building a remote team.

[0:46:55.1] PT: A lot of our learnings about this came from just doing things manually with no code prototypes and just doing things that didn't scale. Moonlight’s first feature was contract projects. We started to see clients using contract projects and then hiring people full-time that they had worked with on the site. We learned by talking to users and doing user studies that they were realizing after working with someone on a contract project that they were a great fit for the team and wanted to move full time.

That's where a lot of our full-time focus has been is contract-to-hire, or what we also call trial-based hiring. We think it's really the best way to build a remote team. I mean, we have a new employee starting on Monday. We've never met him in person. We ran a completely remote hiring process. We remotely collaborated with him. After trying to work with candidates before hiring, I can't imagine going through a contrived whiteboard process without actually doing real work together.

In terms of the future too, this is a really big part of Moonlight, what we're trying to build more into the platform. You'll hear it first here on Software Engineering Daily. If you're trying to go contract at a big company, they typically have very strict contractor requirements. If you're trying to go contract at a company like Microsoft they, have security concerns and things like that, and so you typically need a big insurance package in order to even be able to contract with some of these companies.

We're working right now with Moonlight on being able to automatically provide that, so that you can work with developers through Moonlight on a trial-based process. The developer doesn't need to go out and purchase their own insurance, they'll get a lot of that for free through Moonlight, in order to make that hiring process a lot more simple.

[0:48:39.9] JM: Amazing. I wish we had more time. I have a hard stop in five minutes basically, but we’ll definitely do another show, because there's a ton of stuff I want to get to. Just to wrap up, I see Moonlight as something that is indicative of the changing nature of work. It's one of these little harbingers similar to you look at GitLab. GitLab just doing really, really well. Is a completely remote company and they're developing all these new practices for how to have a healthy company that's entirely remote. Could you just each give me your condensed vision for how the world of knowledge work changes in the next five to 10 years?

[0:49:24.1] PT: Yeah, I think that's a great question. I think that Moonlight does a lot of things that are unintuitive for a remote worksite. For instance, most of the people on the site are based in the United States. We have clients and developers who are located in the same city often and choose to not meet in person. I think those unintuitive things really drive our vision of the future, which is that we think work is becoming distributed and that remote work is really on the rise. We think that knowledge work is going to change the way the world works. There's a lot of different consequences of that.

We've had such a push towards urbanization for the last few years. How does city living where incomes are typically higher, start to change as income is no longer necessarily correlated with where you live, but more on egalitarian principle for a remote team. There’s a lot of different consequences for loneliness in society, for how people continue to get educated. In general, we're super excited about distributed work and the future for that.

[0:50:34.9] EL: Yeah. Just to hit home on those cultural aspects a little more, Philip and I started Moonlight because we were being digital nomads for two years. We were in different cities. We left the Bay Area. We were in places where we didn't speak the language and we were doing remote work for companies that were largely in the Bay Area. We felt that feeling of isolation firsthand and really started to understand how much community is important. If you don't have that at work, which I don't think you need it at work, but you have to have that other outlet.

I think these online and local communities that are based on interests, whether they're inside of work or outside of work, but unrelated from the workplace are going to become so much more important. That's why we think that community is going to be really important within Moonlight itself. Then it also just allows for people who were not otherwise allowed inside of these circles to take part. Part of that is ability to be anonymous, ability to not have to be in a physical space with someone and still feel safe, ability to have an alternative education and still get accepted into a high-paying job and having that just be distributed around the world.

One of the requirements we have on Moonlight right now is that you have at least two years of professional experience. We're trying to figure out ways that we can bring in more junior candidates to engage in the platform. That's just going to require more product changes and banding and things like that, but it will allow for a lot more diverse candidates to interact in this space that has been traditionally not as diverse.

[0:52:12.7] PT: I think that's how the Moonlight network is very different fundamentally from a lot of other professional and community sites is sites like LinkedIn and Facebook have very much been about who you know and using that network to extend beyond that first level to second and third levels. In a completely distributed, global, labor environment, I think our job becomes so much more about connecting disconnected people. I think that's where we have such an interesting opportunity, because there are people that would work really well together that have no close connections and may never have otherwise met each other.

[0:52:49.1] JM: Emma and Philip, thank you for coming on Software Engineering Daily. It's been really fun talking to you and thanks for building a great platform.

[0:52:56.4] PT: Thanks Jeff and thanks for using it.

[0:52:58.6] JM: For sure.

[0:52:59.1] EL: Thanks so much for your time.

[END OF INTERVIEW]

[0:53:04.2] JM: Podsheets is an open source podcast hosting platform. We are building Podsheets with the learnings from Software Engineering Daily. Our goal is to be the best place to host and monetize your podcast. If you've been thinking about starting a podcast, check out podsheets.com. We believe the best solution to podcasting will be open source. We had a previous episode of Software Engineering Daily where we discussed the open source vision for Podsheets.

We're in the early days of podcasting and there's never been a better time to start a podcast. We will help you through the hurdles of starting a podcast on Podsheets. We're already working on tools to help you with the complex process of finding advertisers for your podcast and working with the ads in your podcast. These are problems that we have encountered in Software Engineering Daily. We know them intimately and we would love to help you get started with your podcast.

You can check out podsheets.com to get started as a podcaster today. Podcasting is as easy as blogging. If you've written a blog post, you can start a podcast. We'll help you through the process and you can reach us at any time by e-mailing help@podsheets.com. We also have multiple other ways of getting in touch on Podsheets.

Podsheets is an open source podcast hosting platform. I hope you start a podcast, because I'm still running out of content to listen to. Start a podcast on podsheets.com.

[END]