I'm going to hit record and I'll

  ROB K: it here for us. All right. Hey, Matt. I'm sitting, hello, I'm sitting down today with, is it Matthew or Matt?

  ROB K: Matt. Okay, Matt Ducamp, a software engineer who's been in the industry for close to 10 years now. His most recent position was a senior software engineer at Gusto, a third-party payroll services company. Prior to this, he was employed at Burst, the cloud business intelligence and analytics service, Zeus Inc. and Soasta, which has since been acquired by Acami. Thanks for taking your time today to sit with me, Matt. Other than this brief introduction, is there anything you wanted to correct or add to that?

  ROB K: Well, my last name is pronounced Duchamp.

  MATT D: Yeah, I should even read Duchamp.

  ROB K: He is an S. That's a very common mistake. You get the, that's a pretty common pronunciation, but honestly, if this is in print, then it hardly matters.

  MATT D: Yeah. But yeah, anything else to add to that? Anything you want to say about your background?

  ROB K: I mean, not really. Yeah.

  MATT D: Okay, so one of my goals here today is to provide insight into the life of software engineers. I think one of the best ways to do this is to get a sense of what like, your daily routine is, or, you know, what other software engineers daily routines are, rather than just kind of these, you know, broader summary questions. So to

  ROB K:, you know, could you walk us through a few of those?

  ROB K: Can you tell us a little bit about your work at Gusto?

  MATT D: Sure. Gusto was a hybrid remote work company brought on by COVID, as you might imagine. What that means is that I went into the office only two days a week, two days on Thursdays, and worked from home Monday, Wednesday, and Friday. Besides that, the work was generally the same. There's often a morning meeting around 10 o'clock for stand-up. I don't think that was every day of the week, but I have been at companies that have had that every day of the week, where essentially you give an update on what you're working on, how it's going, whether you're blocked or not, and you can call for code reviews and can someone pair with me later today to help me with this, and so on and so forth. Just make sure that everything's on track. And then after that meeting is over, there will be, depending on the day, you'll either be coding for the next several hours or going into various other meetings. If you're leading a project, very often there's a weekly project meeting about what is going on with that that you'll have to go through. There could be any number of other purpose meetings. I don't know. I'm not sure. I'm not sure. I'm not sure. I'm not sure. I'm not sure. I'm not sure. There is often a monthly retrospective meeting as well that you go to. But typically, after a morning meeting, you're writing code for a while. And when you're done with that code, you put it up for code review. And that involves putting it on whatever your code control service is and tagging some other engineers, typically on your team, to take a look at it. And they'll go through what you wrote and they'll put some comments on it. And sometimes they'll say, yeah, it's all good. Or sometimes they'll say, I wish we could have done this better. Or I actually think that you should do it this way, and so on and so forth. And that'll be kind of a collaborative iteration process. You do that back and forth a couple of times. And when you're done, you merge into the product. And you have to get a proving review prior to doing that. Every engineer needs to have at least one other engineer sign off. It's not like a senior to lower things. I've even had engineers junior to me sign off on my pull request. It's just somebody has to look at it. And so, yeah, it seems good to me. And that's a pretty typical daily iteration process. You'll do a PR once every three days at the least. And I've seen people do it every day, depending on what they're working on. Because you have to have some kind of feature complete portion of code to put in before you do one of those. But yeah, that's pretty typical. Typically, a typical day will involve you working on code for that.

  ROB K: Okay, so there's kind of two directions I want to touch on there. The first one is, so this, as I, you know, I saw, and sorry, my cat is walking through here. Whenever I talk on the phone, it thinks that I must be like, like, losing my mind alone in the apartment and it needs to walk in front of me.

  MATT D: It's obeying its sovereign urge to disrupt your meetings.

  ROB K: But yeah, so, okay, I saw that this is, is this your first senior position that you've had at Gusto?

  MATT D: I think this is the first where you would probably term it as senior. I think part of the reason why I moved on from Burst was that I thought that I should be senior and I was not. So seniority is kind of a hazy concept. I wasn't like a manager or anything, if that makes sense. So yeah.

  ROB K: So, you know, as a, as a senior, I mean, I guess like, you know, it's a little bit hazy. You know, we're talking about like, what is the daily kind of, you know, iteration there and is, you know, has it changed? You know, when you, when you're working at Gusto, is it different being a senior? Do you have to do more things like, are you unblocking people a lot more? Is your blocks of coding, you know, kind of more, or do you have less of them or, or is it, is it more of a title thing at Gusto? scale? A little bit. It's comp... I think Guster can be a little bit different than a lot of other companies, but I do think that senior and basically every rung above software engineer that's still in IC, there's also other words like staff and principal, those mean different things to different people, which is part of the reason why the concept is so hazy. But a lot of it has to do with the scope of the things that you're working on. Instead of working on a widget on the dashboard that has to look and act a certain way, you're in meetings with people talking about how the dashboard should be formed at all and what's the best way of presenting this to our customers. The fact that a widget looks a certain way is like a subtask of the larger task that you're doing. Also a point of seniority is that while I don't manage other engineers, I might be in charge of a project that more than myself is working on. It is my job to, if someone has a question about that project, that I am the person that they go to to get that question answered. I'm supposed to, even if there's another engineer who might even be senior to me working on the project, I have to know what that person is doing, why they're doing it, and generally how it's going. Keeping tabs on all of that is part of being senior, where a less senior position would probably just be working on some defined software feature that we're not. They complete when it's complete. And I think that as you go up, and I see the kind of meeting versus coding load does definitely shift more to the meeting side. But like I said, it's a hazy phenomenon. In some of my days, I had lots of meetings. I would have like four or five meetings. Some days I'd have zero. And the zero days are when I got most of my coding done. Yeah, okay. So, you know, you have your daily stand up on some days. Let's, you know, let's look at one of those days where you do have a longer coding block. You know, as a, as a, like, you know, aspiring junior myself, when I, when I'm like building something, and it's all me, my, you know, my stuff I've built is mainly front

  MATT D:. Like, I'm using GitHub, but a pull request, like for myself, when I commit something, like, because it's just me working on it, I have a feeling that these commits are like, obscenely large, like, I'll make a commit. And it's like, I finished the header, I finished the footer, I finished the body, like, it's all this, you know, what is like a, you know, what does a junior commit look like at a company like Gusto? Um, well, every person has a different committing schedule, shall we say, and I actually, I would say my commit schedule does or perhaps did largely resemble yours for a long time, where a commit was, I'm done working for whatever reason, maybe the day's over, maybe I'm going into a meeting or whatever, and I just commit, I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data. I also like to build systems that can handle large amounts of data.

  ROB K: I see. What kind of software do you specialize in?

  MATT D: I think that's true. You're also designing the button. You don't lose back and forth over the CSS coloring of this particular icon. But you're expected to be able to do that faster and to be able to follow design documents. I have this design document and also a senior you might... I think when I was more junior and I was working on the front

  MATT D:, what would happen is I get a design document, I would implement what was in the document and then return it and then there will be a feedback loop on what I built. I think that being senior also has more of an expectation of, I read the design document, I say, can we change these things about it? I think these things can be done better. Or I think that this might be harder than you think it is. Or I think that this is more aligned with our best practices and to kind of have that level of input and the level of understanding of the product. And then you build with it. When you were... So you said the days when you get the most done are the days when you don't have that many meetings and you get longer code blocks, right? Absolutely. So when you were a junior, your previous positions, were most of your days just, here's my long code block.

  ROB K: More of your days when you're junior are long code block. You're not brought into as many meetings, hopefully. More junior, you might even just have that daily stand up and then you're just kind of expected to code. And then not to say that you can't pair with people. In fact, pairing with people, and by pairing with people, I mean like contacting them with over Slack, like, hey, can you help me code this thing? And we'll just sit together over Zoom or next to each other in the office or whatever and work on it together. That's a pretty normal and accepted thing. And you should definitely do that as much as you can or as much as you need to. But yeah.

  ROB K: Can you expand on that a little bit? So like, you said two people, a pair of programming, you're working together. Does that mean if I'm designing a button that the person next to me is designing the button next to it? Or are we working on the same exact thing just at the same time? MORGAN You are working on, so it can be a difficult process to describe, but you're working on the same thing at the same time. They're not designing a separate portion of the thing. And that probably what you're doing when you're pair programming is you, the person who has asked this person to pair with you, is sitting there with the keyboard and is writing the code. And they're essentially watching you write the code. And maybe advising you like, oh, you missed a semicolon here or whatever. Or they could be like, hey, I have run into this issue. I've run into this problem. I've been working on a problem where I load the product and it crashes. It crashes because of some error that I've introduced in my code. I've looked at my code and I don't know what the error is. The two of you will look through your code together and figure out what is going on with this error. It can be very surprising the value that an extra set of eyes can give your coding. Typically, when I'm doing pairing, I'm either pairing with someone who has asked for my help or I'm asking someone about a specific thing about my code. It's not just like, hey, will you write this code with me? It's usually like I've had a problem or I want a second opinion or I don't know how to do it. Hey, I need to do a thing that I know that you did recently. Can you walk me through the thing that you did? Hearing that process, that would be so helpful, I think, for somebody like me who is basically learning all this stuff through whatever resources are statically out there. It's a little counterintuitive, too, because I think the initial thought is you've got two developers. It's obviously faster or better or more efficient to just have them doing their own thing. But like, OK, so you have these long coding blocks. I want to push back just a little bit on that. So there's actually been academic studies. I believe I have concluded that actually having two developers working together full-time is more productive than both of those developers working individually. That said, though, that is not a strategy that I've ever seen or heard of any company taking full-time because they want parallelization in multiple projects and stuff like that. Developers are typically working on one thing at a time, but pairing is very common. It's just not the norm. My mother is a software engineering professor and I believe she's done work on no actually two people working on the same time is more efficient than those two people working on two different things. I don't know as much of the academics about it as she does.

  ROB K: I mean, just in your experience, too, though, you talked about it a little bit, but why do you think that is where having two people just sitting together produces something that has a little more value to it?

  MATT D: I think there's a variety of reasons, one of which is that you can just kind of get tired working on something or you can just kind of get into a mindset of, I've been working on this code for a while, this is how the code works. And even just getting a fresh set of eyes to go, does this really work the way that you think it does? And that can change everything. And there's also just knowledge differences. Your coworkers know different things than you do. They work with you. You're trying this for the first or second time. They've done it 10 times already. And they're like, okay, I'm going to try this. And then One of my favorite things about software engineering is that I've only been doing that for 12 years and I've focused on packaging. That's when my passion for solver-logging came in. Well, I was looking for talent on server side for anHyb and I found Jack Leo and his team and began working on backend構造newvTools. And that was the reason why I picked software at that time, it's going to take you 30 minutes versus them 30 seconds, but you don't know how long it's going to take them to figure it out for you. And you don't know how long it's going to take you to figure it out for yourself. So, uh, you're, you're, you're constantly weighing that. Do you think that's a, like a, one of the bigger, like higher level skills of a junior is to, to know, or to be able to have a good intuition on, okay, when should I go up the ladder to get help here? It's definitely a skill that you develop over time. Um, I wouldn't expect a junior to come in with the ability to do that. Uh, precisely. Um, I would just advocate that them doing, them doing what I call due diligence, which is if you go to someone senior or not, again, like you can pair with people that are your own level or even below you. Um, if you go to someone and say, it doesn't work, help, that's not super helpful. They might help you, but I always want to go to someone with, this does not work. Here's the problem that I am having. These are the things that I have tried to resolve the problem. And here are the results of those. Then, that pairs down. Instead of them going through a diagnostic process of the seven things that you could try, you've already tried five of them. But here's the two that you haven't. You can jump to that immediately when you're pairing with them and use less of their time and energy on you.

  ROB K: All the examples I've given so far have been fairly basic, like working on a button or designing this component. What was your main code contributions at Gusto? Were they front-end focused or were they full-stack?

  MATT D: I was in a full-stack role at Gusto. That said, though, I've been a front-end engineer most of my career and I was still front-end weighted at Gusto, which is where I prefer to be. I was full-stack because I really want to make sure my skills are up to date. A lot of what I did was front-end. The main thing that I worked on at Gusto is I was on a team relating to a lot of the common code that nobody really owned before at Gusto, particularly the dashboard. On the dashboard, we have these notification objects. We called them to-dos, but they were also called stickies and stuff like that. They're basically like a list of tasks. They were just like a little widget that had some code text in it. They were tasks that we want you to do and we want you to do this, like file your taxes, pay your employees. You need to enter this data into our system so that we can do a thing for you. The idea was when you did all these, you were good to go. You did your job. Everything you need to do, everybody's going to get paid on time, your business is running great. Wonderful. And so this system predated me, but we did extensive updates to this. And we did sorting for them and prioritizing of them and changing the copy and changing a lot of the best practices around doing this. And this wasn't the only thing that they worked on, but it was kind of one of the major things we were working on. And the project that I was working on when I got laid off was adding due dates to these. So like, oh, you need to pay your employees before this date. And we're going to tell you what that date is. And we're going to have it rise up on the page to say like, you really need to do this. This is coming up. And like, no, you should have done this yesterday. It's at the top. Like it's red letters. You have to do this. And what happened to that project after I left? I don't know. But yeah, that's where a lot of my work went. So I do want to still touch on, if we have time, on some of the stuff you just talked about with that individual project and kind of like, get a sense of timelines there. But you did kind of reference it a little bit. So one question I do have, and it's a little sensitive, so I don't want to push it all, but I think it can be helpful for people to hear about. What was that 24-hour period like when you're employed at Gusto and then you wake up and you get some news? Yeah, I mean, I think there's a lot of things Gusto could have done better in communicating.

  ROB K: I see. What about your coworkers?

  MATT D: I'm not a big fan of talking about layoffs. My direct experience is I was in the office on Tuesday and Tuesday was a normal day. I had my normal meetings and I worked on the project that I just described to you. In fact, even for lunch, I went out and I walked around with some of my coworkers just talking about stuff. I'd like to think that they didn't know. I'm pretty sure they didn't know at the time, but in retrospect, it did seem like a last walk. I had a walk with my manager the week before and with them on Tuesday, but I don't think they knew. In fact, actually, one of my meetings on Tuesday was talking about that feature that I just described to you in passing and telling them, hey, we're doing these changes. Be on the lookout for it. Then Wednesday morning, I wake up at the usual time, like 9 o'clock-ish or something like that. I'm not going into gusto today. I don't even think I have a meeting on Wednesday. I think Wednesdays are zero meetings. I wake up around 9 o'clock and I grab my phone and I have a text message from a nebulous number to check my email. That's basically what it says. It says, check your email. I check my email and in my email, there is a document sent to me from DocuSign, which is gusto severance agreement. This is how I find out that I've been laid off. I did not. I did some verification on it. Specifically, I had my work laptop and I opened it up and attempted to log in and nope, it has been remotely deactivated. So I'm like, all right, I have been laid off. And so I read the severance agreement and associated packaging and I

  ROB K:ed getting LinkedIn messages from my teammates over the course of the day and over the course of the several days since. Talking about like, hey, I heard you were affected by the layoffs. I hope everything's going well. I'll serve as a reference as you want. In fact, I actually I went from two references to four or five over the course of that day, just people just volunteering. But yeah, that's how I found out I was laid off. And a couple of days after that, that was on Wednesday the 8th. By the

  MATT D: of the week, I had mailed them back their laptop and associated accoutrements and signed a severance agreement for severance pay, which all things considered was relatively generous. But I told people like I was in the office on Tuesday, you know, you have told me and I don't think I have to read between the lines a little bit. So I don't know if I want to like don't put this necessarily on record, but I don't think my manager I might not have known before my manager did that I got laid off because I'm not sure he would have found out before he actually liked me so much. I've gone through a lot of microservices but I've also been involved with machines as well. I actually

  ROB K:ed off as an employee of a company called Ubiquiti, which was kind of like a jewelry store in the south side of California and

  MATT D:ed up

  ROB K:ing my professional career at Ubiquiti. I had to actually send the second thing because there was a tag in my car to get into the parking garage. But I had to send them. But yeah, I just kind of got let go from gusto and took some days off and then

  ROB K:ed applying for jobs. Yeah, and I mean, you know, I think there is a kind of, you know, a normal thought that there is an inherent negativity to something like a layoff. But I don't really think that has to be the case. I mean, it makes sense that it is. But, you know, given a couple more days now, a couple weeks here, I mean, how are you feeling now? I mean, obviously, it was shocking and upsetting to be laid off. But, you know, I didn't, I was not particularly worried. I was already kind of aware that gusto wasn't in the greatest place. And I was already thinking about leaving. In fact, I told you I had two references before I got laid off. Well, that's because I was talking to people about like, hey, I'm thinking about updating my resume. I think the most surprising part about it is that it was a random Wednesday. I had a project planned through the

  MATT D: of the month. I had a program that was going on for longer than that. There was work that was talked about happening in March and April. Performance reviews occurred in November and were going to occur again in May. We were nowhere near any of those. It was just a random Wednesday. In fact, they were obligated to pay me my salary through the 14th. They did. They could have continued to employ me for an additional four business days for zero money. But they did not do that. I'm going to get another job. I've been laid off before. I know there's a big round of layoffs in the industry right now, but there's always so many jobs in software. It might take a few months. It's going to take a bunch of grinding and applying to places. There are things that I would rather be doing than applying for jobs. I'm trying to find time for myself in this. I'm going to get another job. I'm an engineer with seven years of experience and experience a lot of relevant technologies. That's always something that is useful in Silicon Valley. Especially now with post-COVID, a lot more remote work has opened up. I can apply to places like New York City, Los Angeles, middle of the country, nowhere. As long as they're paying a competitive salary and benefits, then maybe I just work from where I'm sitting right now. I'm still figuring that out. I'm reminded actually of what I was told by a recruiter, not recently, but when I was originally applying for Gusto. I've long since forgotten his name, but he said that the important part is not to have job security, but career security. That this is the way the tech industry works. There's layoffs, there's companies go up and down, they just stop existing, they get bought, they get fire-sailed. If you have career security, it's going to come from your skills, your work ethic, and your mindset. If you have career security, you can work for a very long time. It's just going to be for a bunch of different companies. There's not this thing in tech where you get into a company and you work there for 25 years and then you retire with a pension. That doesn't happen. I share your optimism. I'm getting a lot of questions, pushback from people in my life asking me, Hey, how can you be, this is the worst time to be doing this? I share the idea there, which is like, I think there's always a need for skilled developers.

  ROB K: I see. What kind of software do you specialize in?

  MATT D: I work mostly on backend systems and infrastructure. I enjoy building scalable and reliable systems that can handle large amounts of data. I enjoy building scalable and reliable systems that can handle large amounts of data. I enjoy building scalable and reliable systems that can handle large amounts of data. I work mostly on backend systems and infrastructure. I enjoy building scalable and reliable systems this. And there's lots of different ways of doing that. Like I've worked in payroll, I've worked in business intelligence, I used to go to a dating site, you know, and my skills were relevant to each of these companies. So it's really just what type of company you want to work on and what kind of problems you want to be solving. And I don't want to take too much more of your time here, but I mean, you were talking about one of the reasons you wanted to be a full stack or into full stack is you wanted to keep your skills sharp. You know, looking ahead here, are you, you know, where are those skills, like where are the skills going right now? Do you, are you looking towards full stack still? Are you looking towards front

  MATT D:? I think I'm most comfortable in like a front

  MATT D: weighted full stack position, kind of what I had at Gusto, a little bit of what I had at Burst. I like working on the front

  MATT D:. I think I prefer working on the front

  MATT D:. I think I'm comfortable in that. I think the languages and the frameworks there agree with me. And I do like having a product that I can load and look at and iterate on, which is not always true with the backend. It's kind of a lot of component or not component design, there's a lot of systems design on the backend. But at the same time, I think the backend skills are useful. There's more like the front

  MATT D: is just completely dominated by JavaScript and the backend use a lot more, you know, Java, Python, Rails, the occasional C++ even. And just having skills in that give you a lot more opportunity. Like I can work as a backend engineer if I got a good job, good company, but we want you to work on the backend, like, okay, I can do that. Now, where perhaps I would not have been able to do that three years ago. So it is about your skills are really what's going to get you in the industry and keep you in the industry. Because you're going to be moving jobs and doing new things and learning new things all the time. And, you know, here's a question kind of just for myself, but I know there's a lot of people like me looking for jobs right now, especially in the front

  MATT D: too. So I, you know, I work with, obviously, I'm like using JavaScript, HTML, CSS. I don't, you know, I made the choice not to dive into a framework and, you know, I'm reevaluating that choice. But look, my thought was JavaScript is the foundation of all these tools. Let's just focus on this. But I mean, what are your thoughts here? You know, do you think it's worth the time to pick a framework essentially?

  ROB K: Can you give us a little bit of background on what you do?

  MATT D: I specifically picked the framework React out of the air. If you want frontend industry recommendations, if you're working in the frontend industry right now, you're working in TypeScript, which is a strongly typed version of JavaScript, and you're working in React. And the second potato to React is Angular. And if there's a third framework, then maybe you might see someone use it sometime someday. I've heard of VueJS or something like that. I've heard one company using that. Basically, everybody's working in React and the people that aren't working in React are working in Angular. Do you have to know these to get hired at these companies? No. In fact, I actually learned Angular, one, on the job, and two, actually, I learned it working on coding challenges set by jobs I was applying to. So you can definitely learn it on the job. You can definitely learn React on the job. If you want to polish your skills to the maximum before applying to frontend positions, minimum learn React. If not, also TypeScript. Sure. Yeah, that's good advice. Because, again, I've just been kind of waiting to pick a framework. Pick React. Yeah. And that's what seems like the most popular out there. Okay. I mean, I don't have...

  ROB K: I don't want to take too much of your time here, Matt. I have other questions that I could ask, but I think I outlined a 30-minute interview here and we're a little bit over that. Is there anything else that I didn't ask that you want to put out there to talk about at all?

  MATT D: Well, I kind of already generated some soundbites, I think. I'll say that I've worked with people that have come into software from other industries before. And the way that I met those people and the way that they broke into the industry was through a bootcamp. I cannot remember the name of the specific bootcamp those people used, which would probably be very useful information. But I do know that they did work in an intensive bootcamp. It took them a couple of weeks and they were coding a full-time 9 to 5, 5 days a week for 2 weeks, 10 days type of thing. They learned relevant technologies and they got hired at jobs to be on the same level as me. Now, they were the same level as me as I was 3 years ago, but they got hired. They were software engineers and I presume that they've been promoted by now. So this is a thing that people can do. You don't need to have a CS degree, although having a CS degree does certainly help. But if you don't have a CS degree, I would recommend a bootcamp.

  MATT D: And a couple of those, that's another group of individuals who I'm going to be interviewing here, Hack Reactor graduates.

  MATT D: Oh, yes. I think that sounds familiar.

  MATT D: There's a couple other ones that... I saw about five, but I think Hack Reactor was the largest out there. I definitely have looked at doing a coding bootcamp. I think at this point, I've spent enough time self-teaching to where I think it'd be worth it, but I think the cost is the limiting factor for me. They range to about $15,000 to $20,000 to do a 14, 15, 19 week bootcamp. That might have changed and they have a lot of finance options there too. But at the

  MATT D: of the day, bottom line, it's about $15,000.

  MATT D: Well, I spent $185,000 getting a Bachelors in Computer Science. I don't really know that much about a bootcamp. I just know that these people did get hired.

  ROB K: I spent $60,000 getting a Philosophy degree. So, sometimes I'm thinking about that too.

  MATT D: You're still out ahead.

  ROB K: I was a copywriter for a while and it was okay. But I think diving into at least the front

  MATT D: for now, and I have interest in the back

  MATT D: too, but I think it

  ROB K:s with breaking in a little bit.

  MATT D: Yeah, I think front

  MATT D: is probably easier to break into than back

  MATT D: anyway.

  MATT D: Well, anyway, I think... That pretty much wraps it up for us. I'll go ahead and turn off this recording here, Matt.

  ROB K: Cool. Sounds great.