I’m here today to talk to you about Empowering Early-Career Developers.

In this talk, we’ll cover why empowering your team is important, the process my team has been using, and strategies that you can use to empower your early career team members.

My name is Mercedes Bernard and I’m a senior software engineer with Tandem in Chicago.

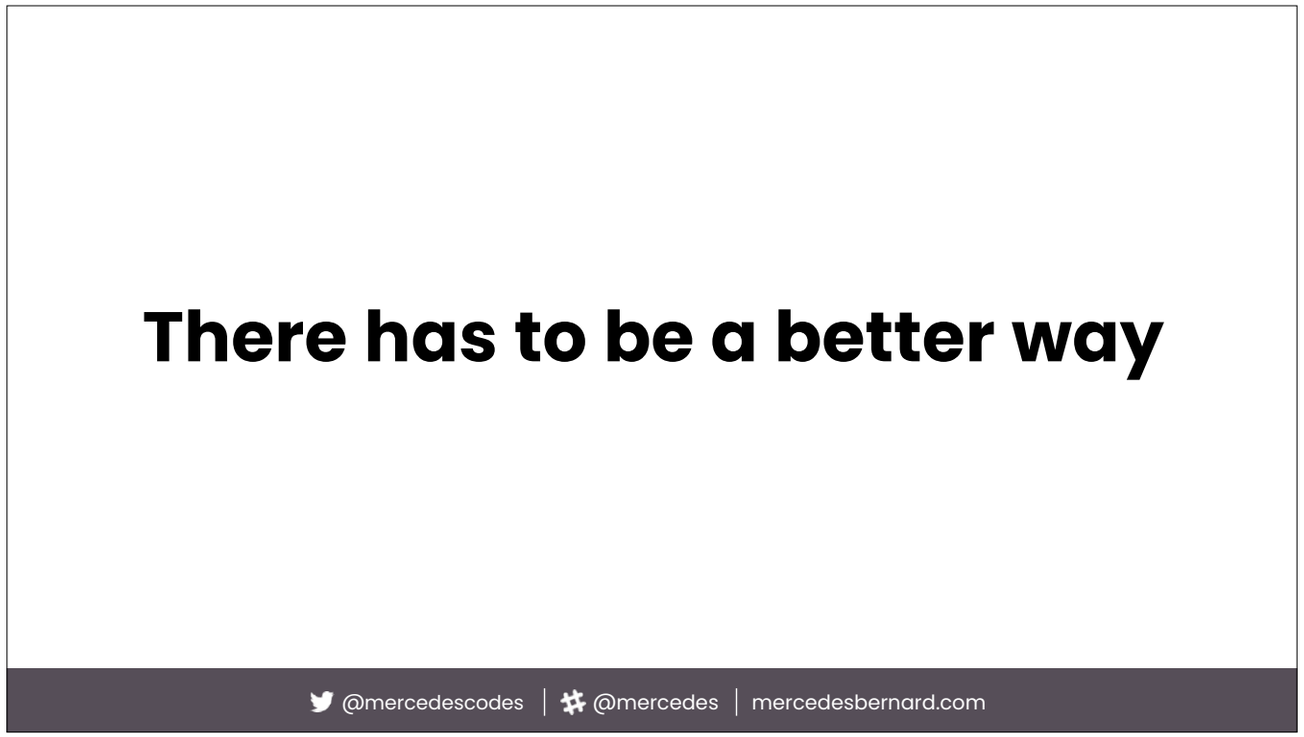
I want you all to think back on your career progression. How did you get to where you are today? How did you grow from intern, apprentice, or entry-level to your current role?

Did it feel a bit like trial by fire?

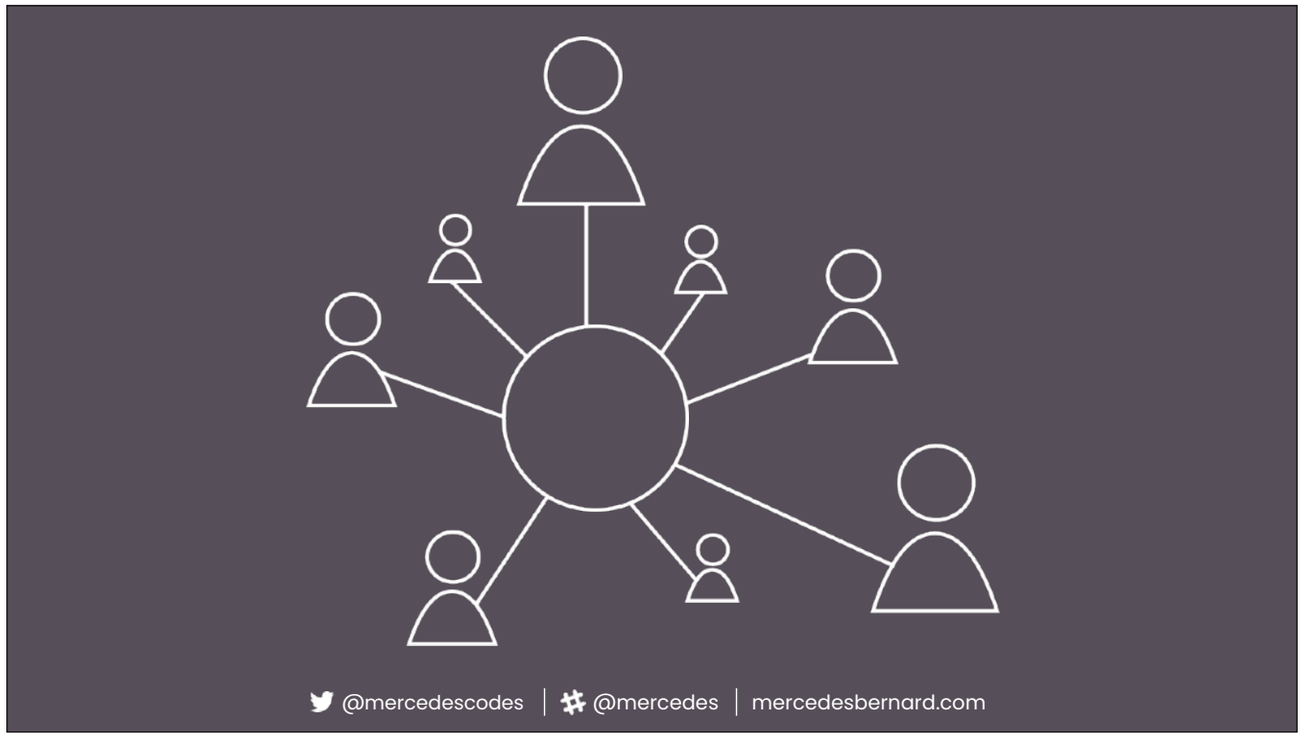
Or like drinking from a waterfall?

My experience definitely did. I can remember a few instances in particular like when I was a just-hired consultant, still in college, and I had to explain to a very angry client why deploying untested code to production at 1 in the morning is a bad idea. Or another time at my first job out of college where a large payment transaction project fell in my lap with a poor estimate made by someone else, and I had to try to manage unrealistic client expectations on my own.

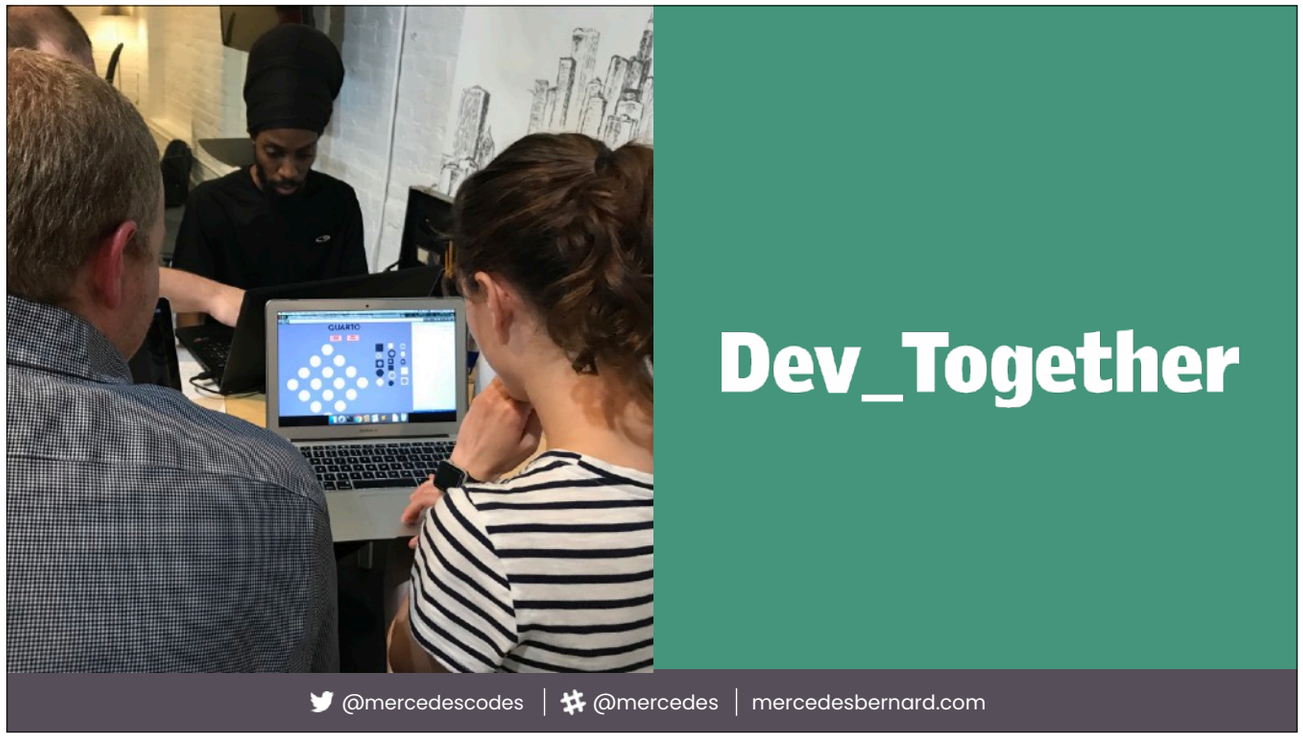
In both of these cases, I did not have support of someone in a senior or lead position.



And unfortunately, I don’t think my experience is uncommon. There has to be a better way.



In my current role, I’m leading a team of 8 developers. This isn’t the first time I’m leading a project, but it is the first time I’ve led such a large team of varying experience levels, and where I’ve shared responsibility for their professional growth.

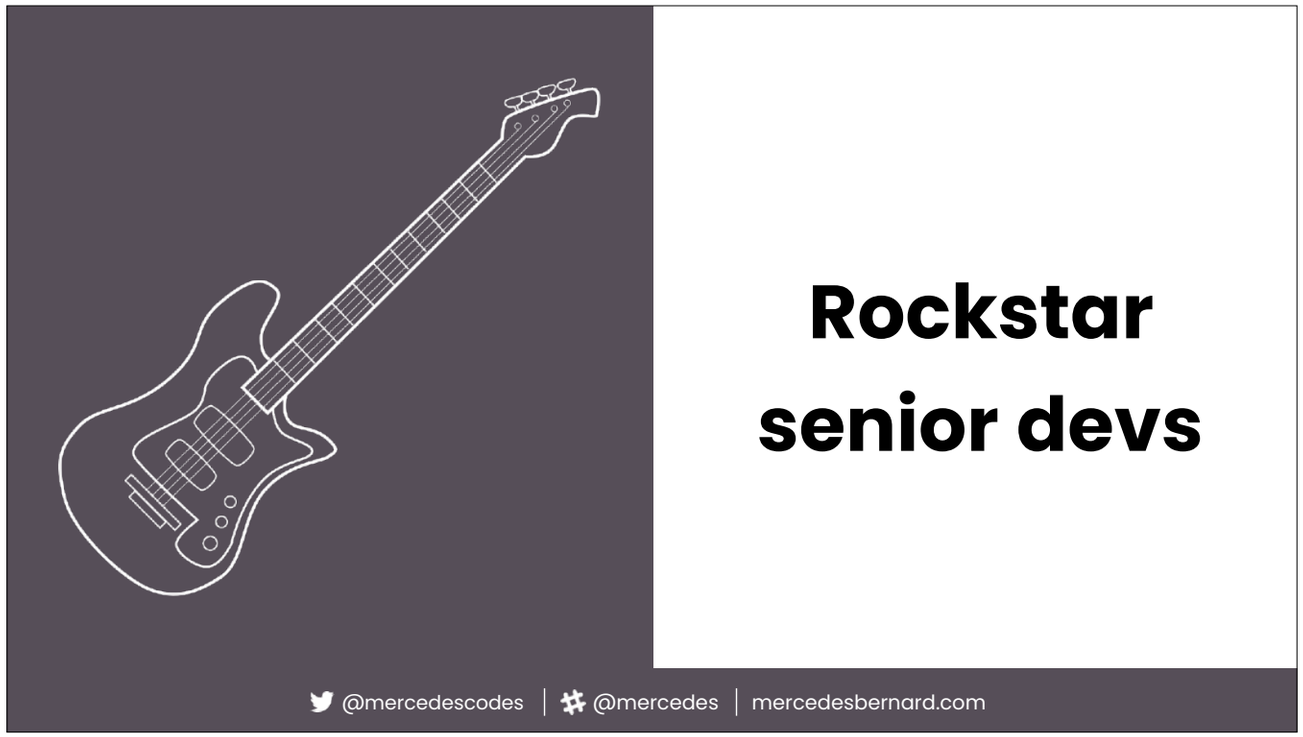


I’ve also started a community in Chicago called Dev Together that connects people starting their development career with technical mentors. Both of these experiences have opened my eyes to the tremendous need in our industry for solid mentorship and coaching for those early in their careers.

The market problem

This brings me to the problem with our current technology market.

Companies fear investing in junior talent right now.

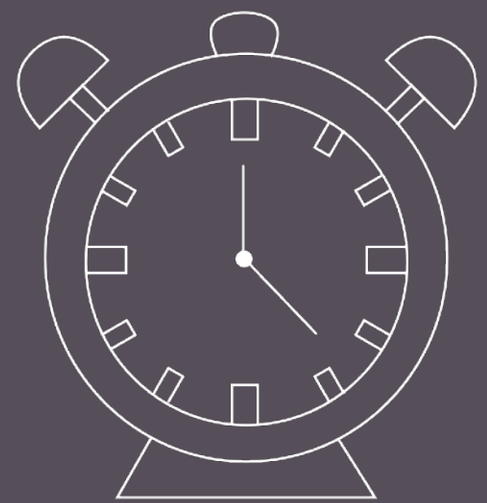


If you look at any of the job boards, everyone is trying to build out teams of rockstar senior devs. In my opinion, this is a losing strategy.

It costs so much money to have a team of senior engineers. Recruiting costs, their salaries, the attrition caused by lack of career growth (when everyone is senior, where is the growth potential?).

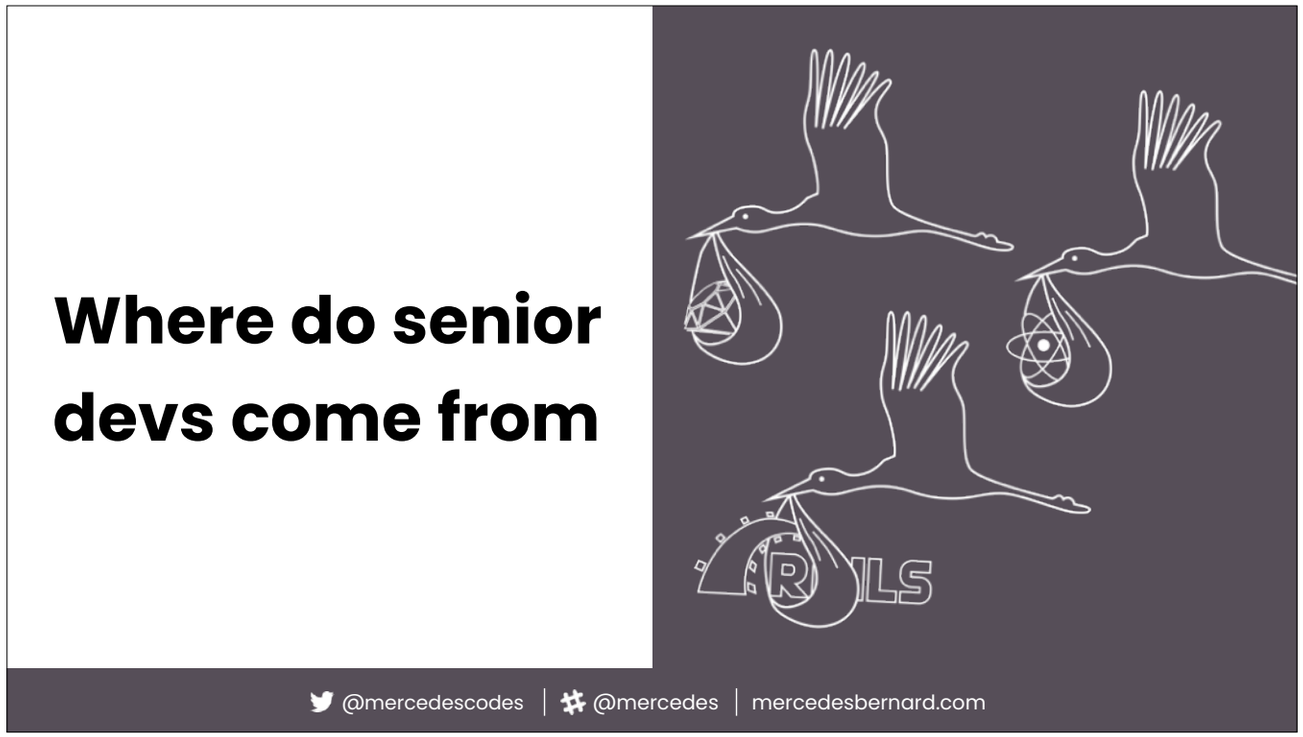
And not every task on a project requires a senior's experience. Companies are overpaying for work that someone early in their career could be challenged by and learn from. In my opinion, it costs less to mentor early career devs than to hire a team of all seniors.

Seniors don’t have time



And the argument that seniors don't have time to mentor your less experienced employees isn't valid. Your mid and senior employees need experience mentoring and coaching if you want to grow them into amazing team leads and managers that teams want to work with.

It’s an unfortunate reality in this industry that job hopping is the fastest and easiest way to level up. And I understand why this would make some companies hesitant to invest in their early career talent. They worry that after they spend all this time, money, and energy, their early career devs will just leave to get that mid or senior title.



But where do you think senior devs come from? I was talking to someone recently who made the joke of the ‘senior dev stork’ and it just doesn’t exist.

People who feel valued, who are challenged, and who have the support of their leads and management are more likely to stay than those who are given the “intern work” or busy work until they get bored and leave just to do something different.

So I’ve been reflecting on how best to offer mentorship and coaching to meet this need. We need to empower our early career talent if we want to retain them and grow them into the senior devs we’re all looking for.

~~Junior~~ Devs

And the second part of this talk is “Early-Career” I’m sure you’ve noticed by now that I’m staying away from the term “junior.”

Early Career Devs

Why is reframing ‘junior’ as ‘early-career’ important?

“Junior” has a negative connotation. I don’t think its warranted and I don’t think its earned but there are many who view junior as ‘probationary’ or other types of less than rather than viewing it as someone who is learning and growing.

Identifying someone as “junior,” because of the negative connotation, can cause imposter syndrome flare ups. If you use the term junior and the developer thinks you might view that as less than, it makes them feel less than.

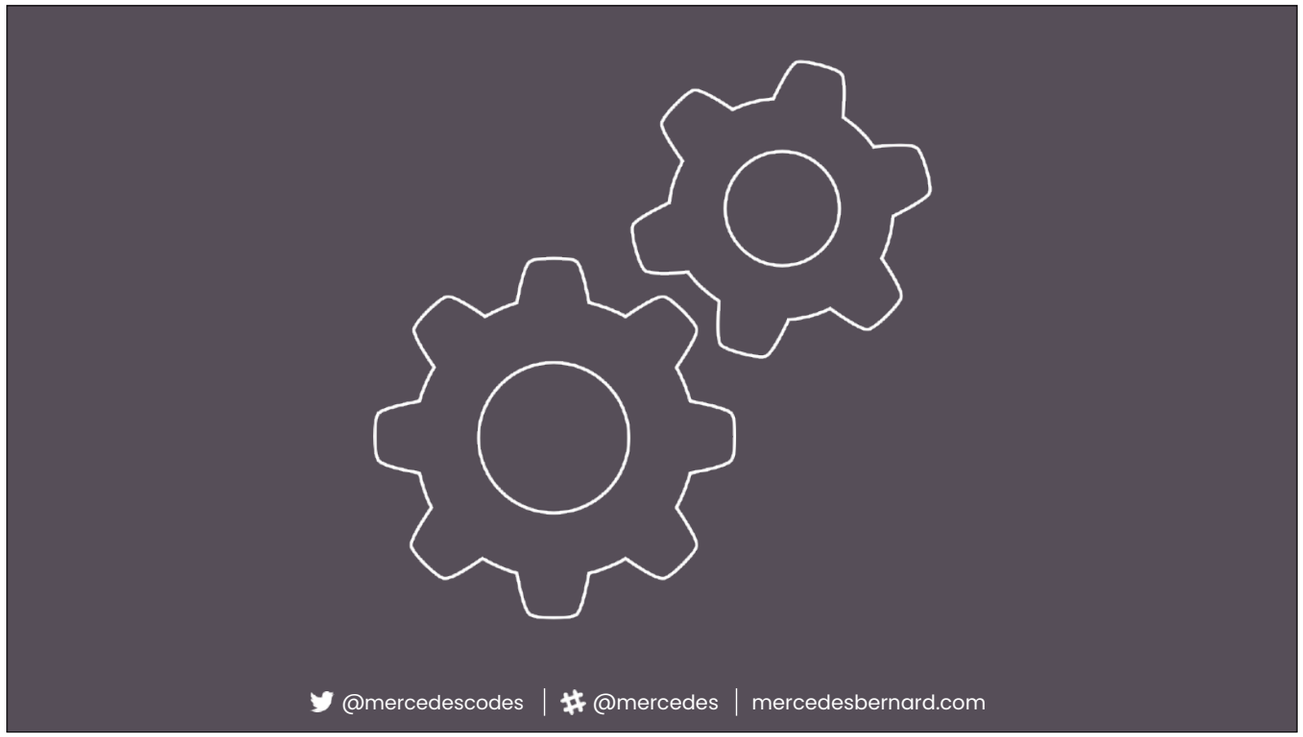
Just because this is their first development job, doesn't mean that they're "junior." Career switchers have so many transferable skills, often the communication or soft skills that can be most difficult to learn on the job.

Early career devs have fresh perspectives to bring to your team and no bad habits to unlearn.

In my experience, early career devs have all of the enthusiasm and none of the arrogance that comes from a lot of senior technologists.

Early Career Devs

Shouldn’t we use positive language when talking about them instead of labeling them as “junior”?



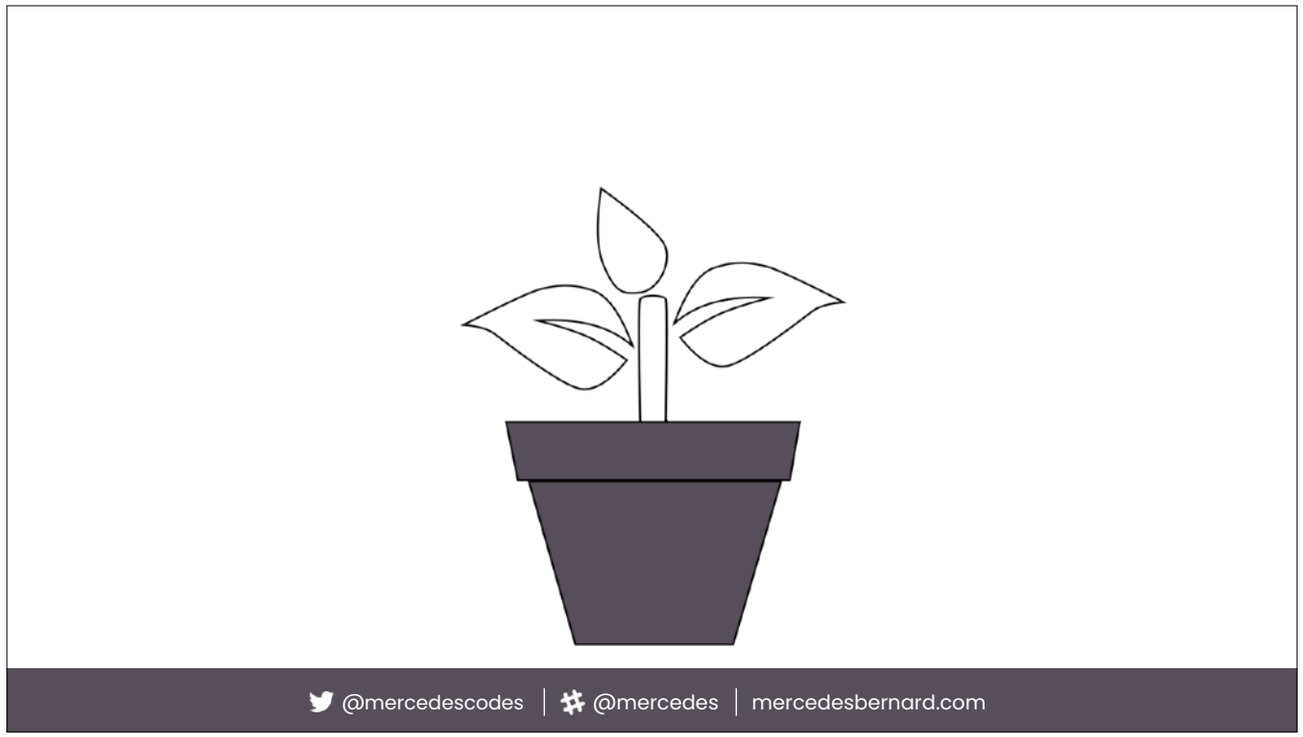
Let’s shift gears a little bit and talk about senior devs for just a minute.

I’m sure everyone has ideas about what the characteristics are that define a senior or lead developer. For me personally, I break the defining traits into 2 categories of skills and experience.

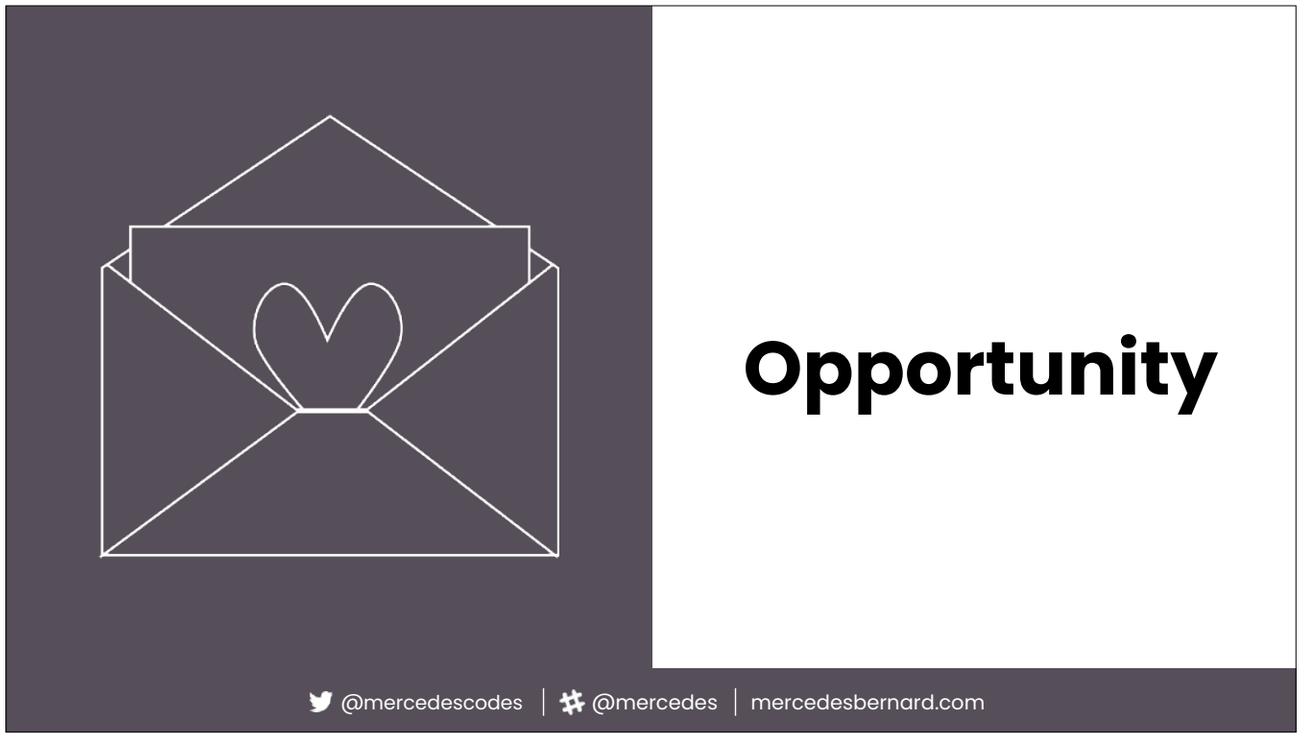
For me, a senior dev has strong technical communication, time management, meeting facilitation, self-direction, consulting, and client engagement skills.

And a senior or lead dev has experience making decisions, being accountable to those decisions, looking at the big picture beyond the feature checklist, and leading a team.

Your list may look different than mine depending on what kind of development you work in or what you value and that’s totally fine!



So how do you help foster growth for someone in their early career to a senior dev who exhibits all of these traits?

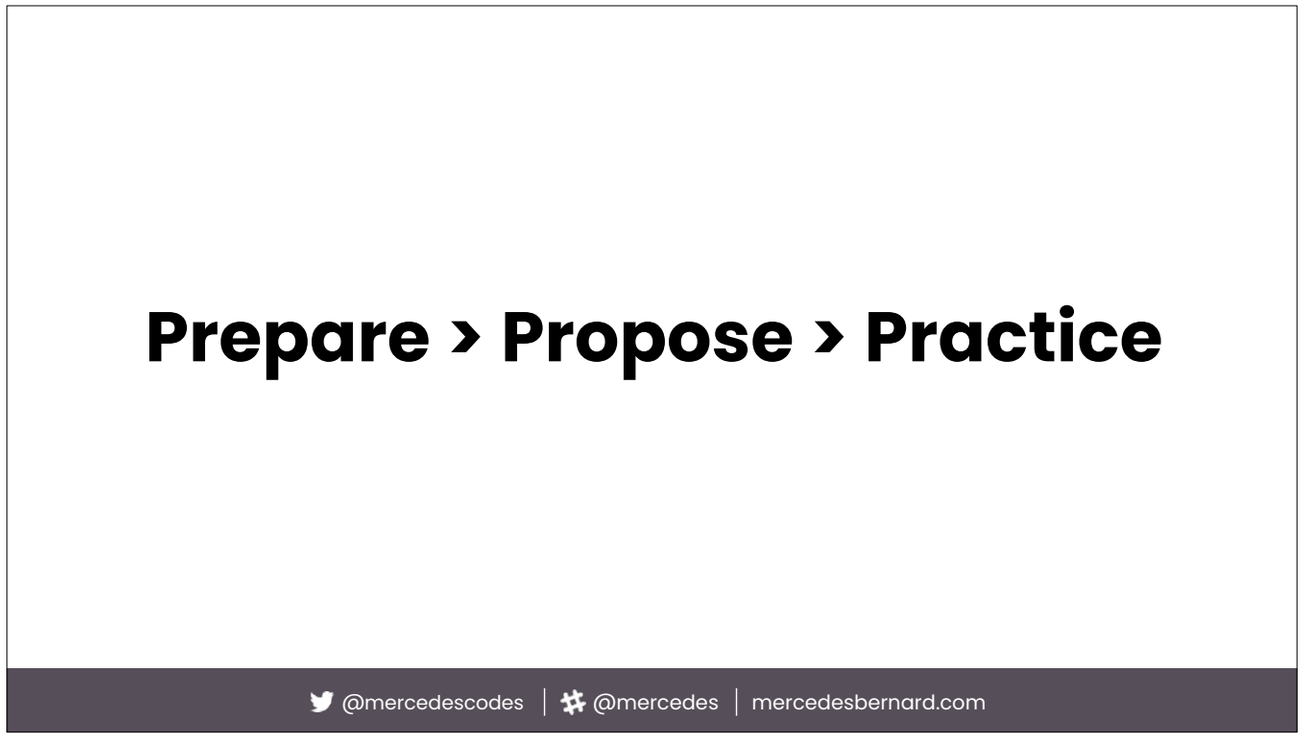


You need to give them the opportunity to do all of those things. Before they have that role. You need to have sustainable practices in place to grow those skills.

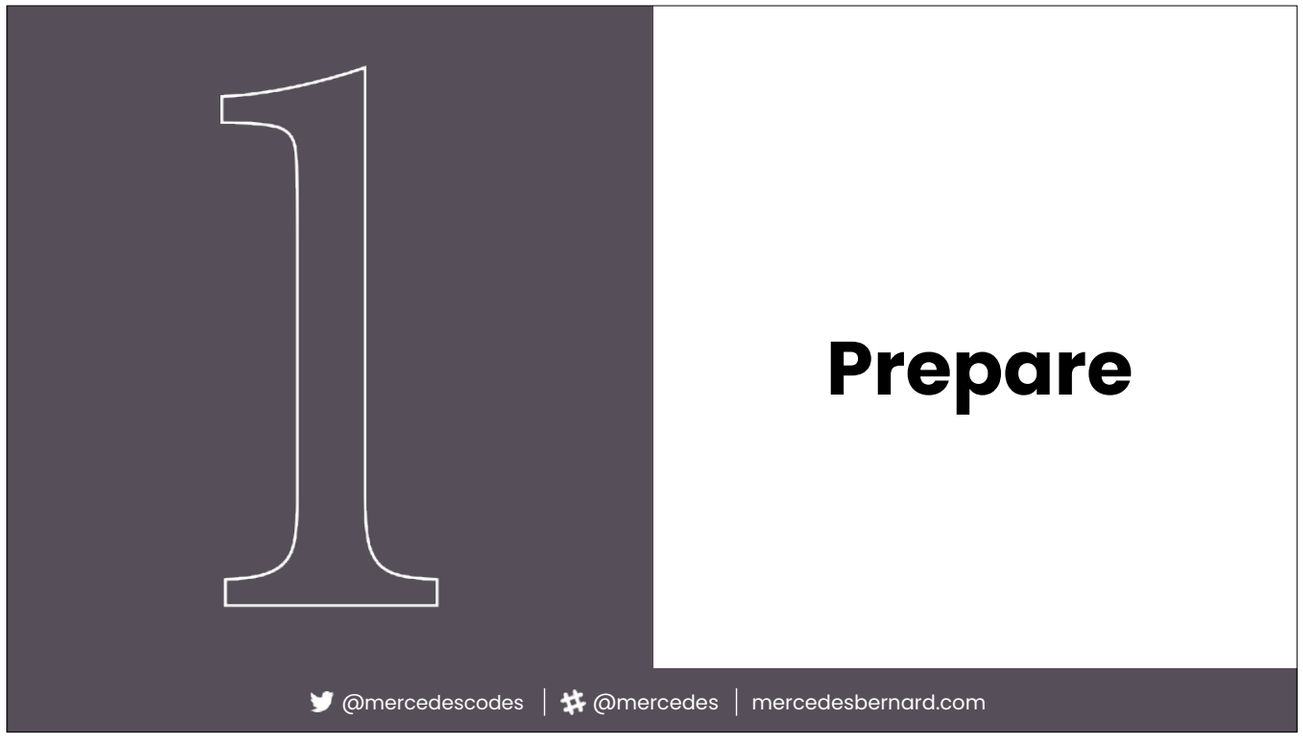
Remember, we don’t want a trial by fire experience for them the first time they step into a lead dev role.

Case study

On my current project, we've been putting processes in place to facilitate these types of opportunities. I'm going to walk through what we've done, what we've learned, and how you can bring something like this to your team.

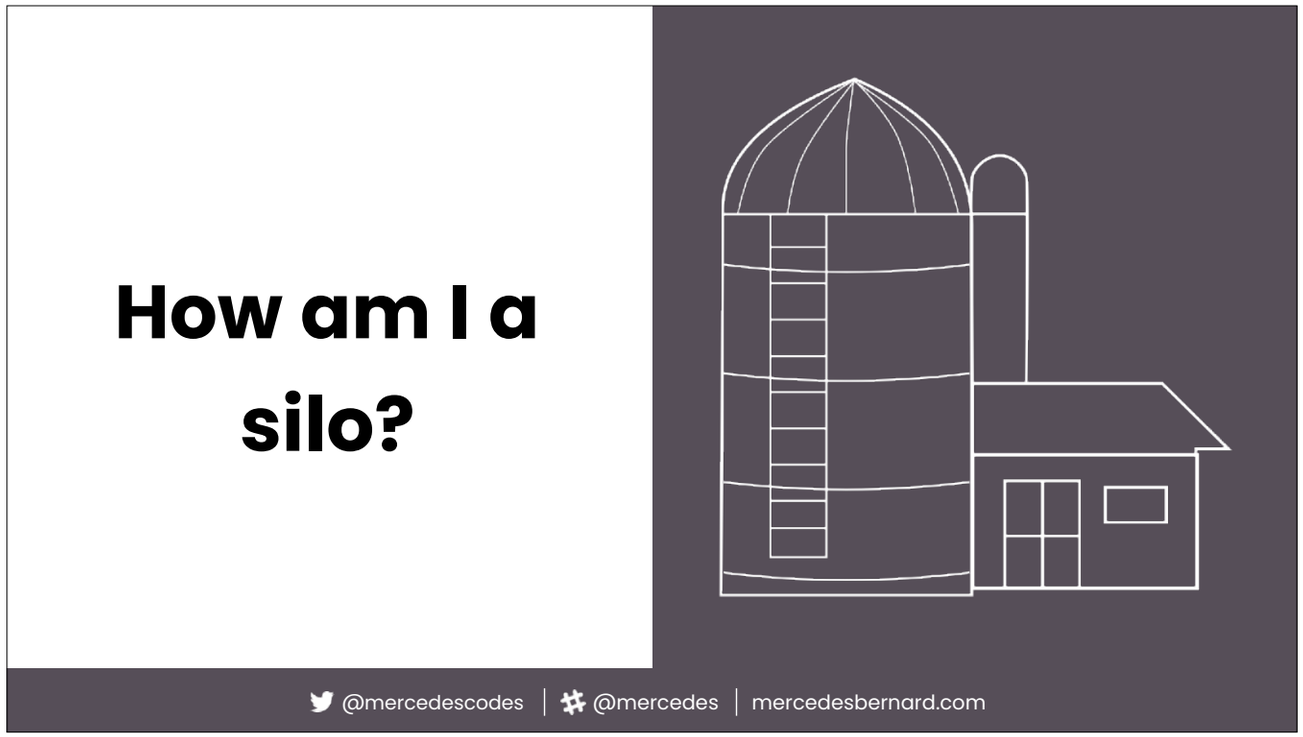


Putting these processes in place kind of followed three main steps, Prepare, Propose, and Practice.



The first step, Prepare, is where this plan came to life.

Working with my manager, we were talking about ways we could mentor and coach my team to help their professional growth. We thought it would be great to find a way to give them some ownership over the project as well as space to practice skills that they weren't getting a chance to practice much.

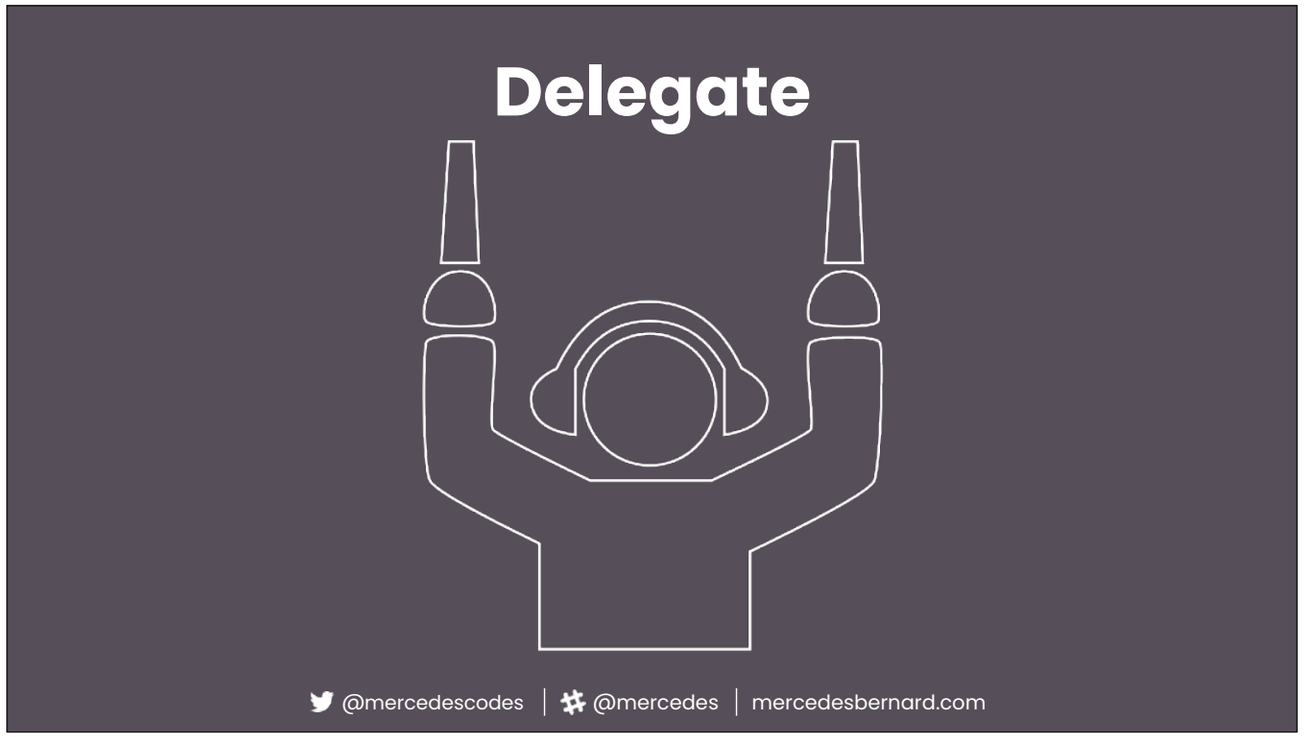


So I started by asking how am I a silo? What roles and responsibilities do I have that the team doesn’t get insight into and that results in them not getting the experience they need to grow?

For our project and my team, the answer to this was client communication. I owned almost all of the technical communication with the client. My team could really use the opportunity to practice and grow their consulting skills but didn’t have a place in which to do that. I was also leading all of our decisions and meetings, but many people on the team were ready to take on some of that responsibility.

After reflecting on that, I spent some time clearly defining the purpose of what we were trying to do. I wanted to define the roles and responsibilities so that everyone had clear expectations of the new practices we were putting in place.

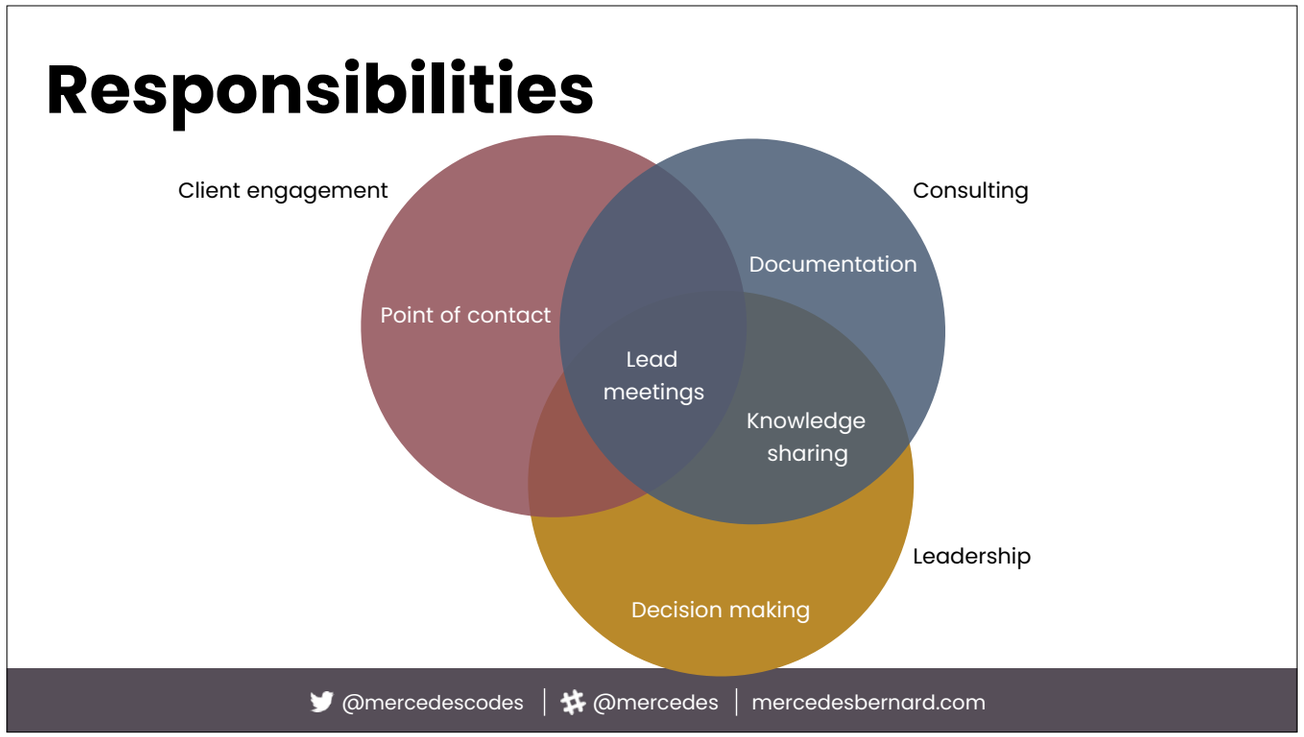
I decided to have each member of our team own a feature area of the application.



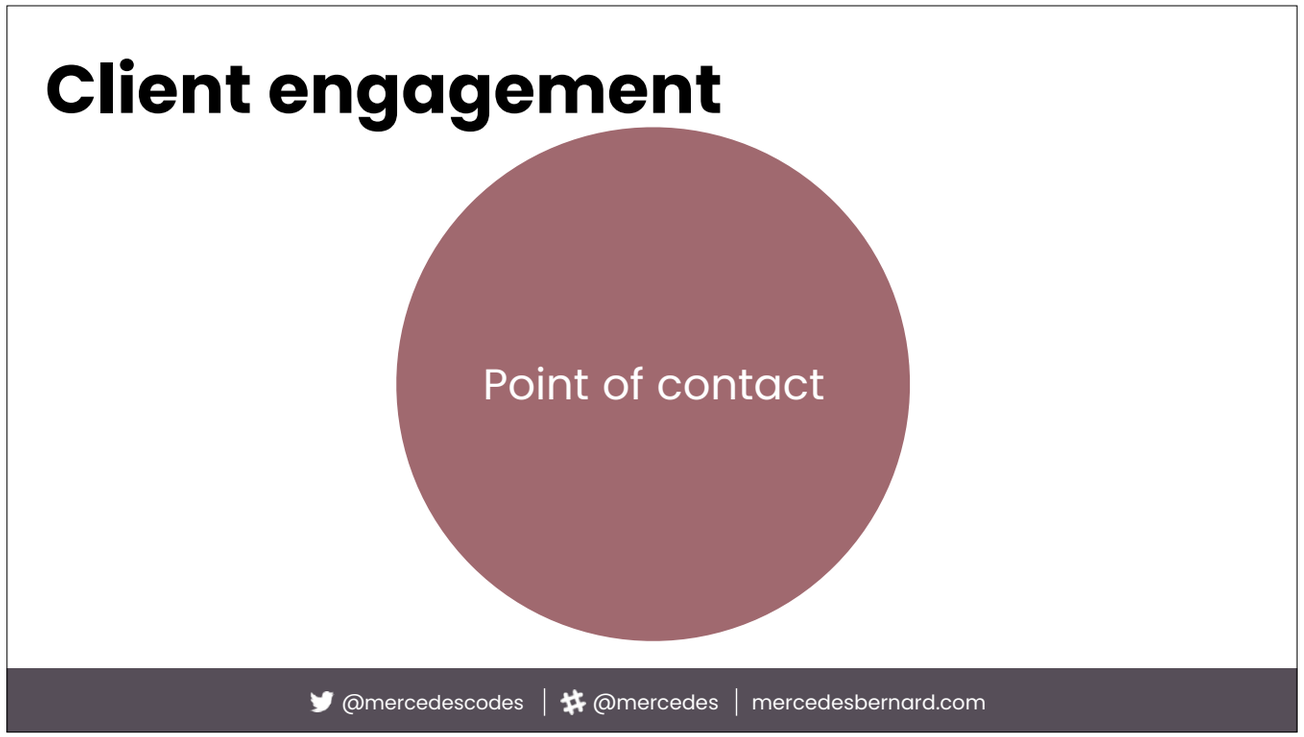
And finally, I had to prepare myself to delegate. This was the biggest challenge for myself personally and I think for a lot of people leading a team. You need to prepare yourself to share responsibility and accountability, and to let go of some of the micromanager tendencies that tend to come when you're leading a development team.

To connect to a jet-way bridge, an arriving plane must center itself perfectly on a white "J line" painted on the tarmac. Because the airplane door is behind the cockpit and the J line is underneath, pilots can't hit their marks perfectly without help. The Federal Aviation Administration actually requires three ground crewmembers to guide in each big plane — a "marshaler" under the nose of the aircraft and two "wing walkers" on either side.

The second step to implementing our new process was Propose. In the Prepare step, it was mostly me and my manager coming up with ways to put a plan into place. But in this second step, I wanted to introduce the plan to the team, create a shared understanding, and get feedback from the group about the roles and responsibilities.

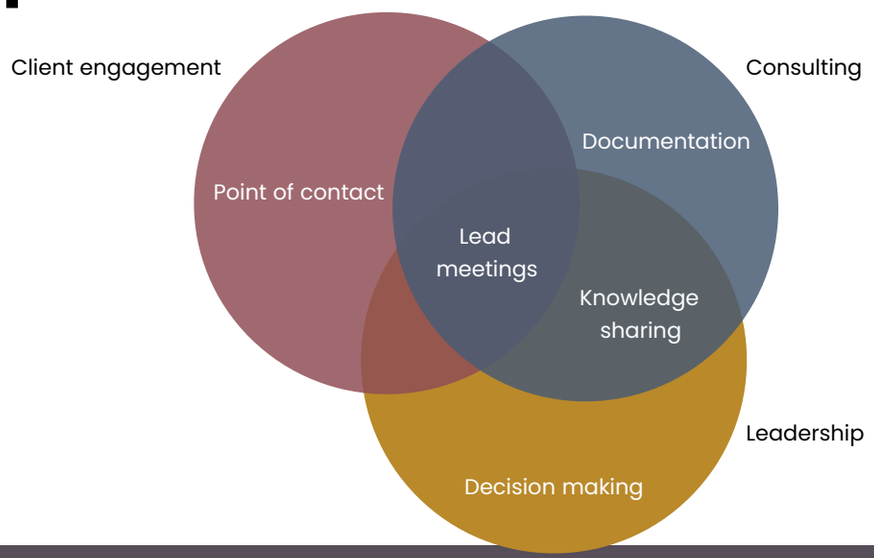


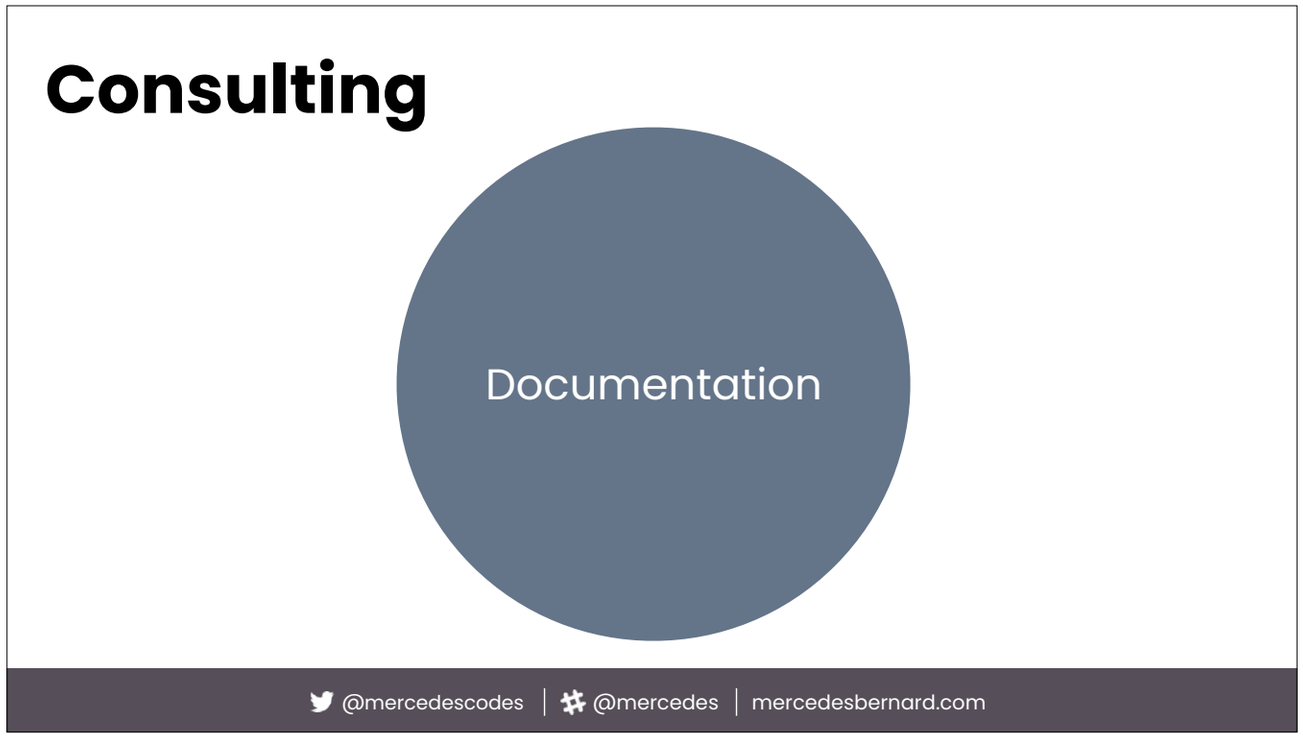
These were the responsibilities for feature ownership that we came up with for my team and we were able to discuss, clarify, and refine as a group. These might serve as ideas that you could potentially use.



The first part of feature ownership is being the client's point of contact for that feature. As a feature owner, the team member facilitates communication relevant to their application area with external stakeholders and they are a trusted expert for them. The team member gets to practice asking questions of clients, holding them accountable to communication deliverables, and answering their questions about features, timeline, and priority.

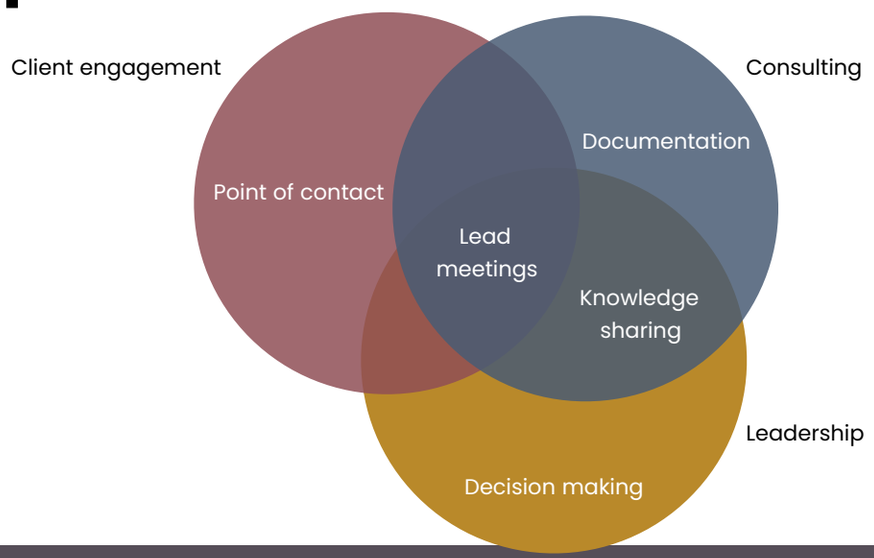
Responsibilities

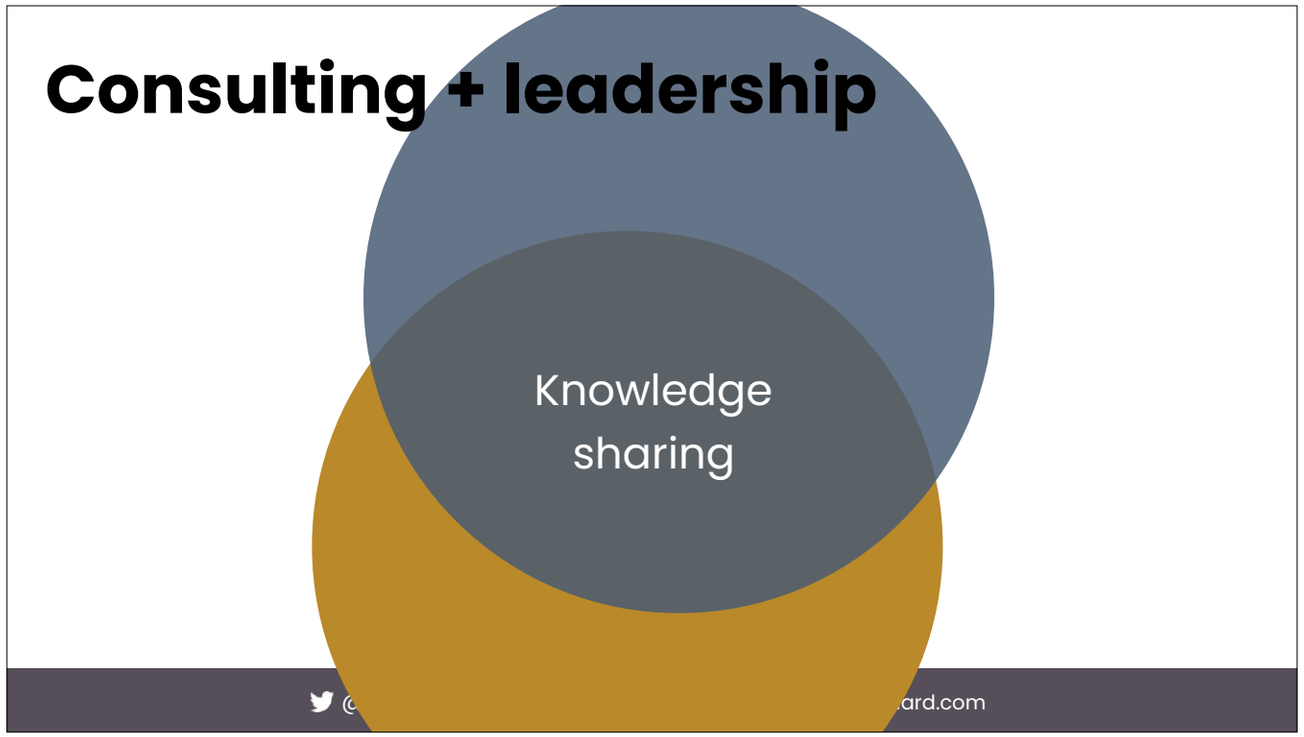




Next, the feature owners need to document the system, user stories, access, needs, etc of their application area for the team. This isn’t a flashy part of consulting, but its important. Good documentation and written communication skills are often underrated and everyone has a tendency to oversimplify the amount of effort and skill it takes to write good requirements and user stories.

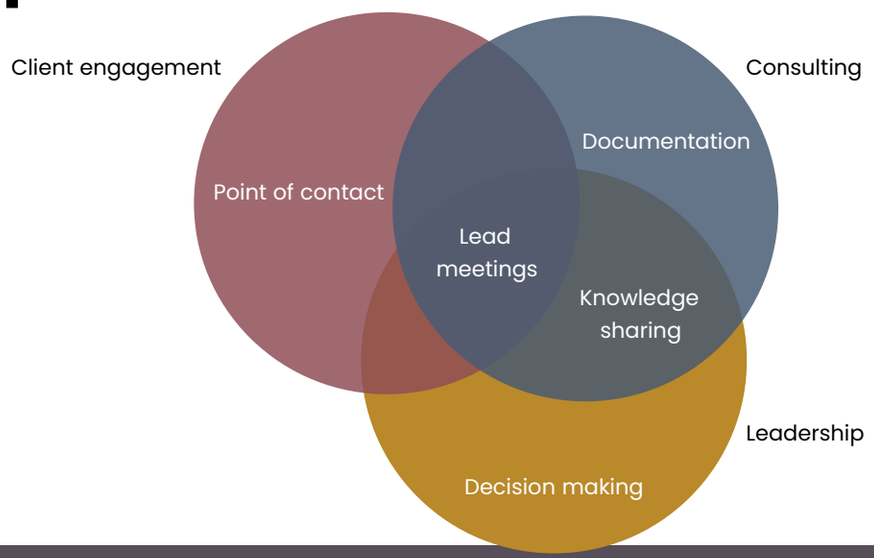
Responsibilities

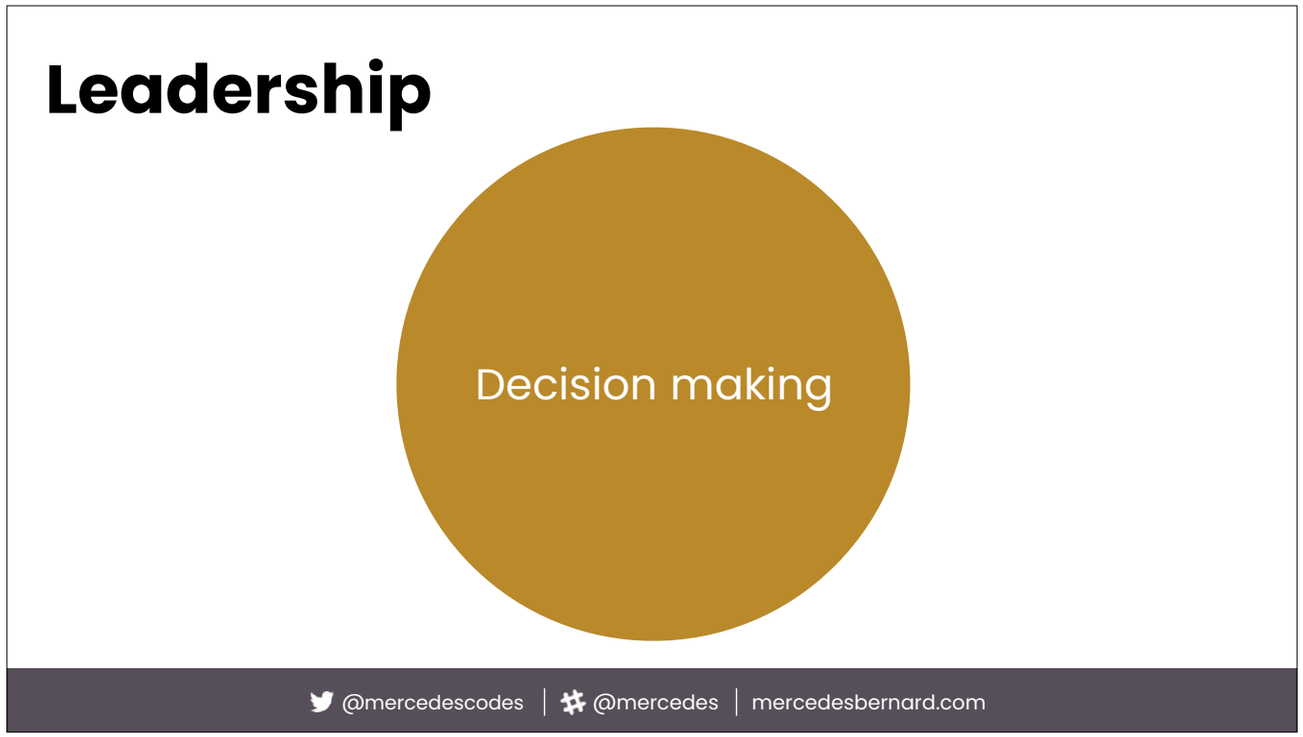




A natural progression of documentation is knowledge sharing. The feature owner needs to make sure to communicate learnings to the team. We don’t want this consulting work to happen in a vacuum and we definitely don’t want knowledge silos, so its important to share what they’ve learned with the rest of the team.

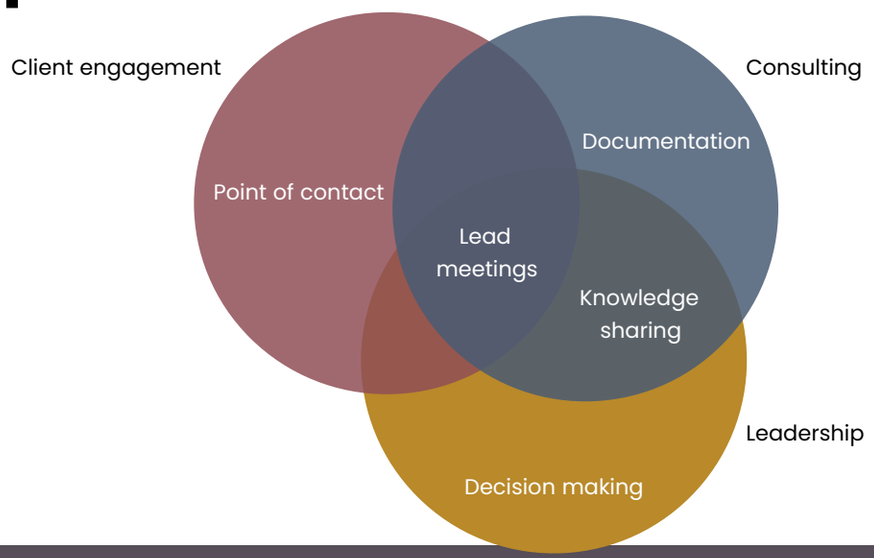
Responsibilities

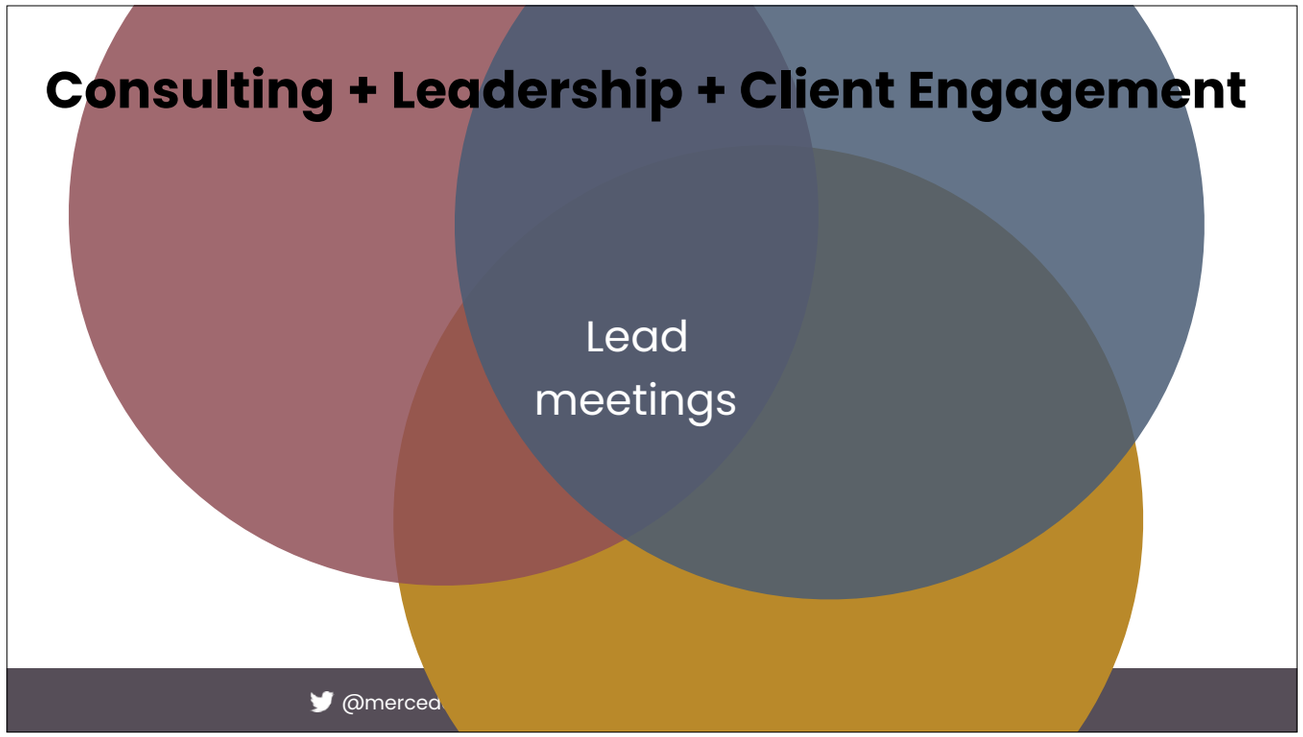




The feature owners are a key stakeholder in implementation decisions related to their application area. They get to practice making and being held accountable to their implementation decisions. And they get practice advocating for their decisions.

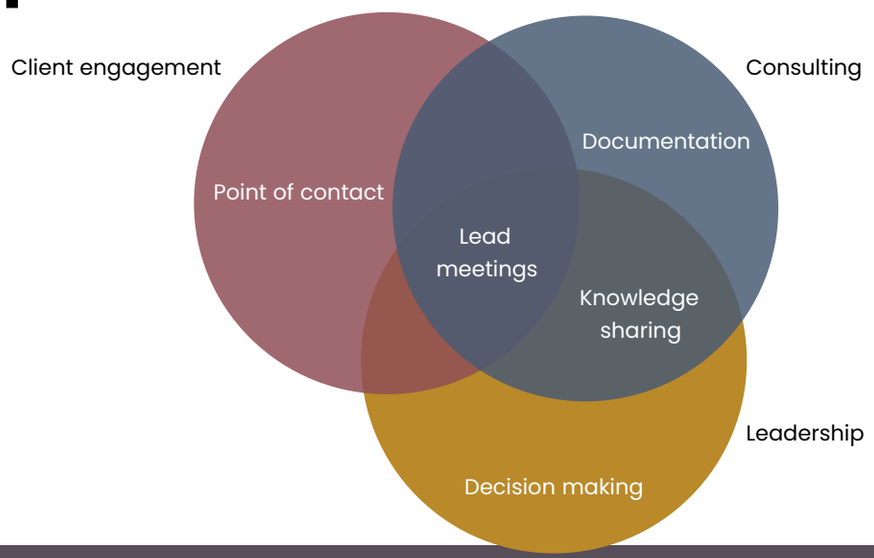
Responsibilities



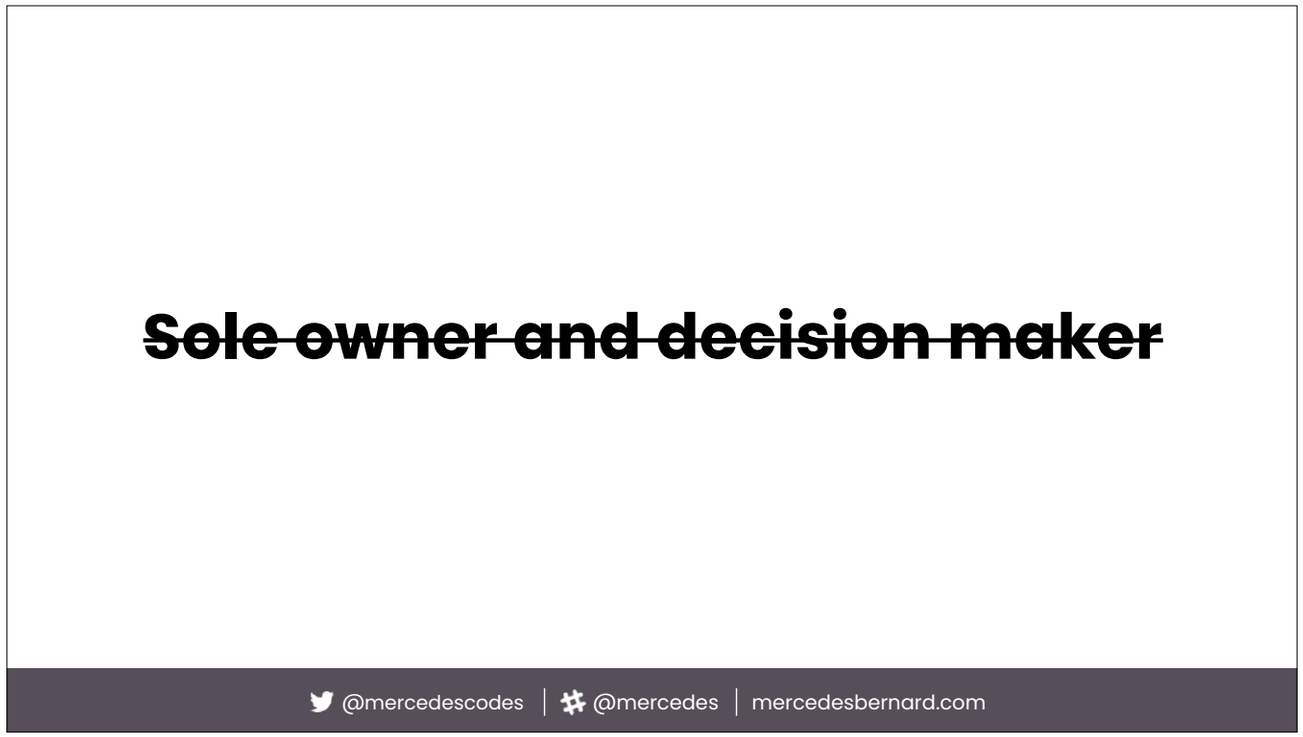


And finally, feature owners attend and lead meetings related to their application area. This helps build domain expertise, facilitation skills, and confidence. It also had the added benefit of helping the team learn what makes a meeting valuable and how to make sure the meetings you schedule are valuable.

Responsibilities

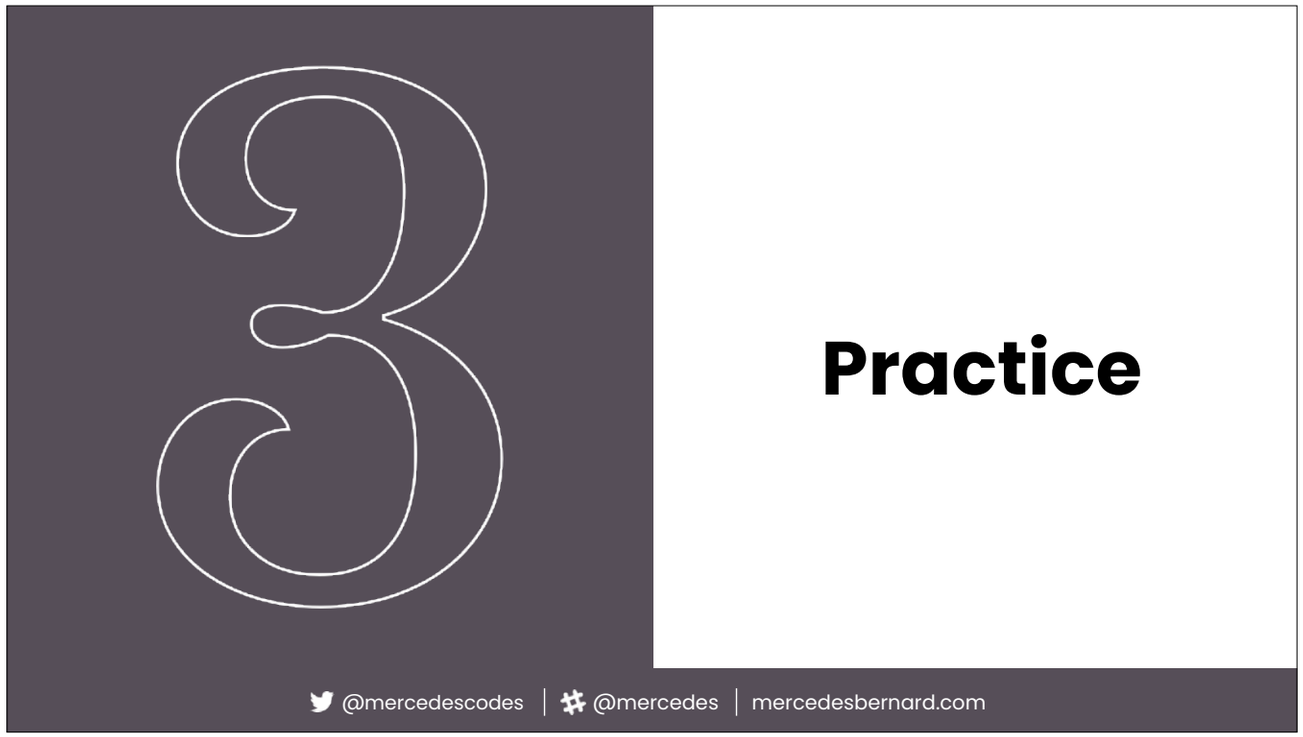


Sole owner and decision maker

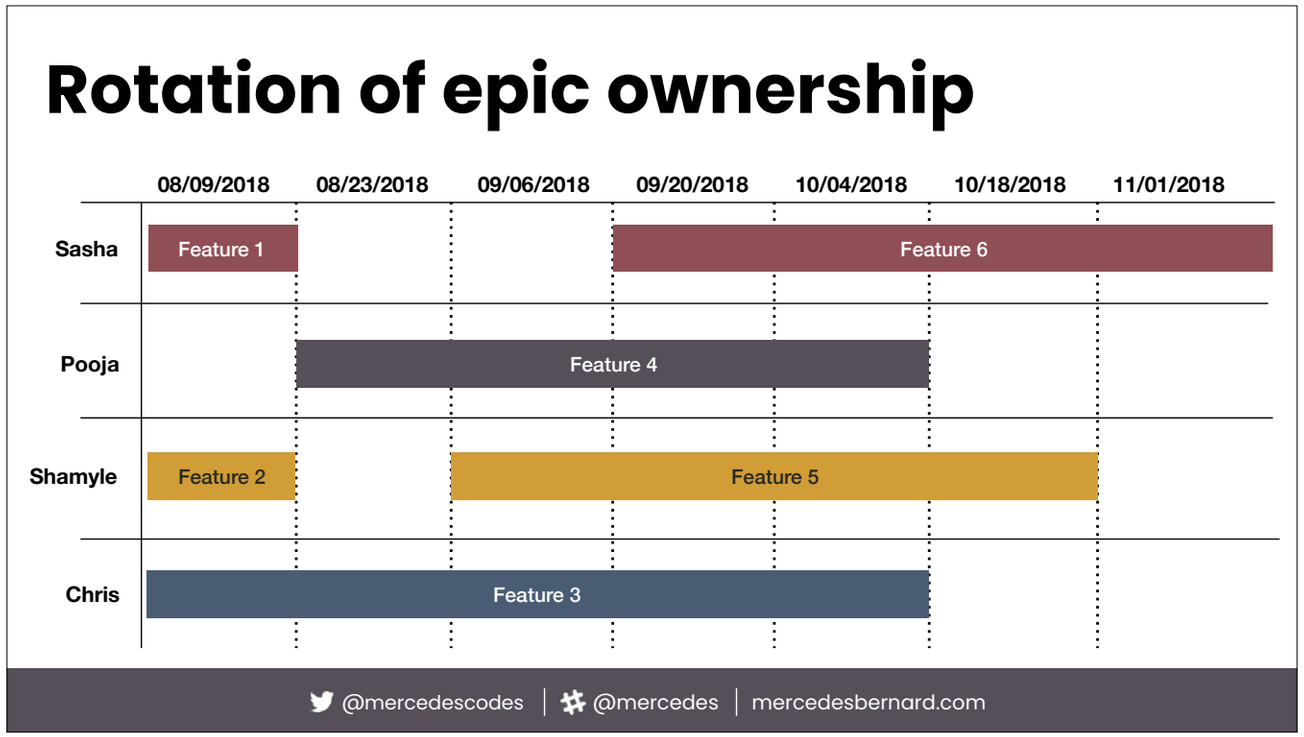


During the Propose step, I thought it was important to make sure my team understood what this new process **was not** as well.

They wouldn't be the sole owner and decision maker of their application area. They would still be supported by me and other senior engineers on the project. They'd share decision making responsibility with both me, our design lead, and our product owner. And they would still need to answer to the team because they wouldn't be the only one implementing the stories and features in their application area 🙂 just like a team lead.

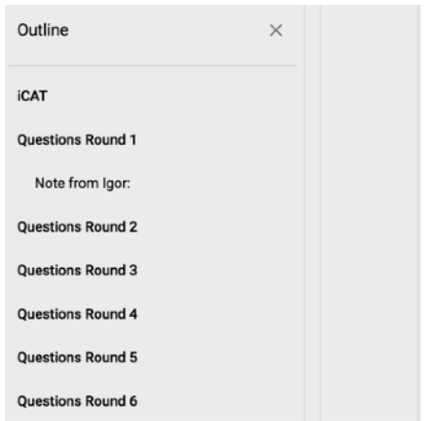


The third and final step for us to put this new process in place with everyone sharing these responsibilities was to Practice.



We have an ongoing rotation of feature ownership as we progress through our product roadmap. A few sprints before we are set to start working on a new area of the application, I assign the area to one of our team members who reaches out to the appropriate stakeholders with our client to gather technical requirements (API integrations, file formats, job schedules, etc), understand their present day workflow, and start identifying any risks as we learn more about that particular application area.

Documentation

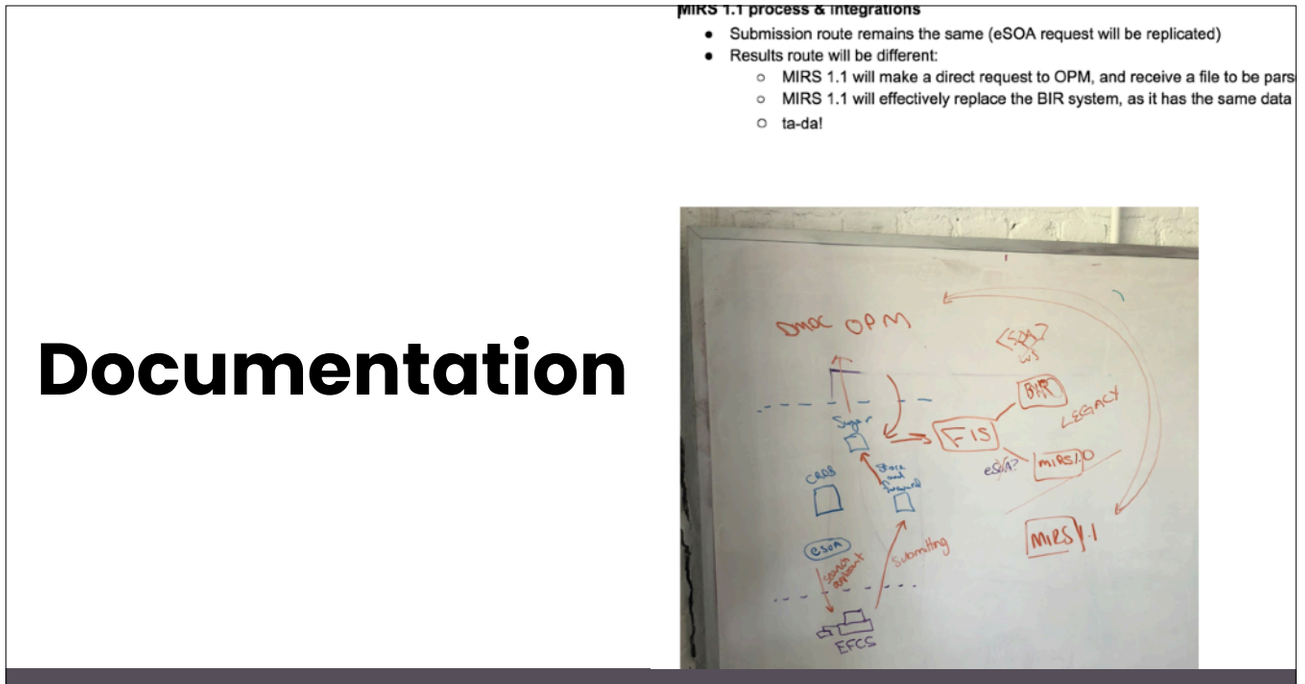


Questions Round 4

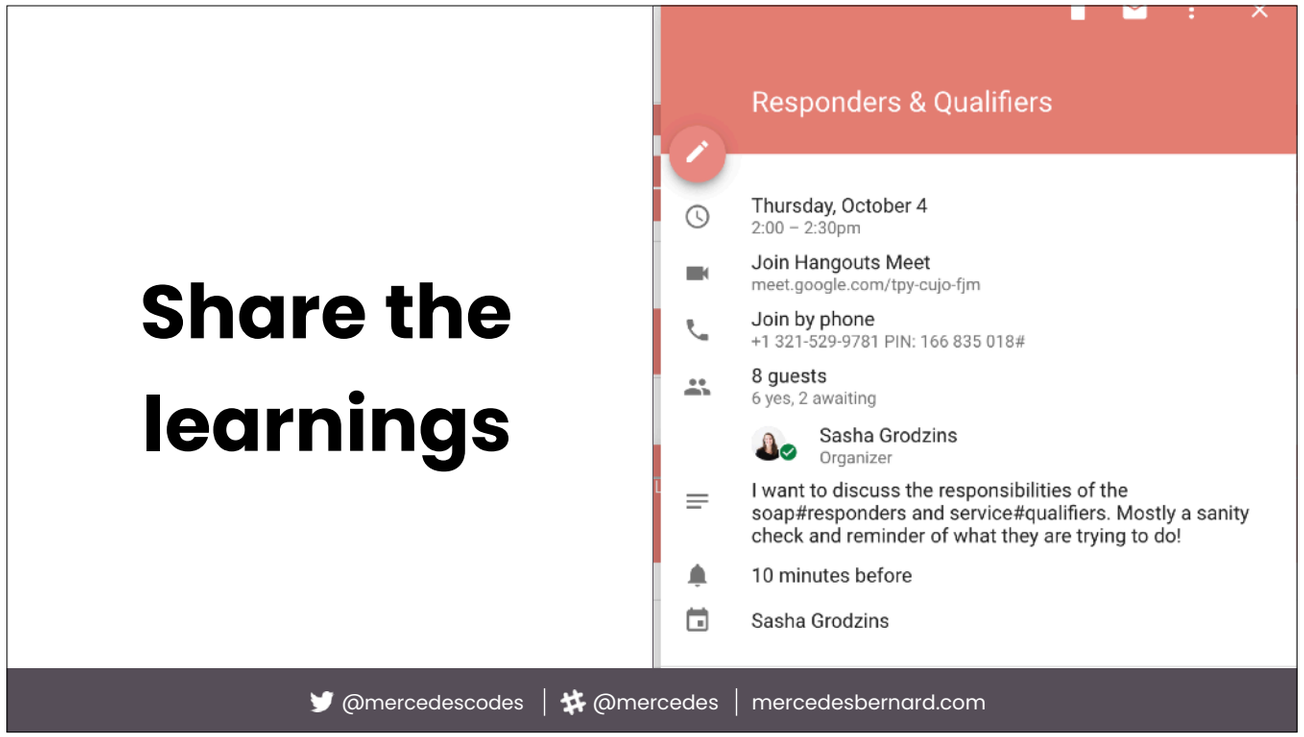
- Is there a response to the incoming iCAT payload?
  - Yes. Sample SOAP responses attached.
- Do the iCAT payloads come in one at a time? Do they come in batches?
  - One at a time.
- Do they come on a schedule?
  - No. Applicant takes a test. DPAC's ICAT system scores it.
    The resulting record gets queued up on their end to be transmitted to us.
    If our system is not accepting transmissions, the record goes back into the queue to be re-transmitted few minutes later…
- Can an applicant have multiple payloads?
  - Not in the same SOAP message.
- Is the Test Administrator identification a DOD ID or SSN?

Here is an example of the question document that Chris has been tracking. You can see that he's been tracking each round of follow up questions we've had and puts the answers in there.

Documentation



Here is some of the documentation Sasha has put together. We had a discussion about some legacy architecture and what we needed to keep around for backwards compatibility. She took a picture of our whiteboard and put just a few bullets describing our conversation.



They also distill all of the learnings into what the team needs to know (ignoring the extraneous stuff that isn't important for the work at hand) and coordinates the best way to share that knowledge with the team.

Sasha scheduled a meeting for the team to discuss the responsibilities of some of the classes in our code as a reminder for how we should be using them.

Then they start writing out user stories for the team to start working on whether that's to do more research into complex requirements or outstanding design tasks that we learned about through our requirements gathering or the actual implementation stories with details if we're ready for those.

Here Shamyle has created a sub-task for us to make a SOAP request to a third party service and he's added some implementation notes for whoever picks up this card as well as attaching a sample request.

Continues dialogue with the client

Thank you Laura and Jason!

Just to recap, these are the two questions we still need answers for:

1. We are planning to show some of the information that we receive back from DMDC, and also have a place to upload the DD214. If there is a discrep separation date, what should happen?

2. Who should be allowed to view Prior Service information?

And a third question also came up:

3. What should happen if a code from DMDC is invalid? For example, what if, in DMDC's response file, the Interservice Reenlistment Code (Interserv-

Thanks so much!

Throughout the duration of the feature implementation, the team member continues to facilitate the dialogue with the client and get answers to any questions that may come up as well as field their questions and feature requests.

In some cases, this is just following up on questions the team has.

Continues dialogue with the client

Hello everyone,

Hope your week is going well. I just wanted to follow-up on the items we discussed last Friday. As a reminder, we discussed gathering the following by tomorrow:

- Business logic rules for validating an ASVAB test
  - What makes a test valid / invalid?
  - When can an applicant retest?
- DLAB layout files
- Server address for sending the nightly files to DPAC
- Verification of the nightly file schedule

Thanks

And in other cases, this is holding the client accountable for their deliverables.

What are we learning?

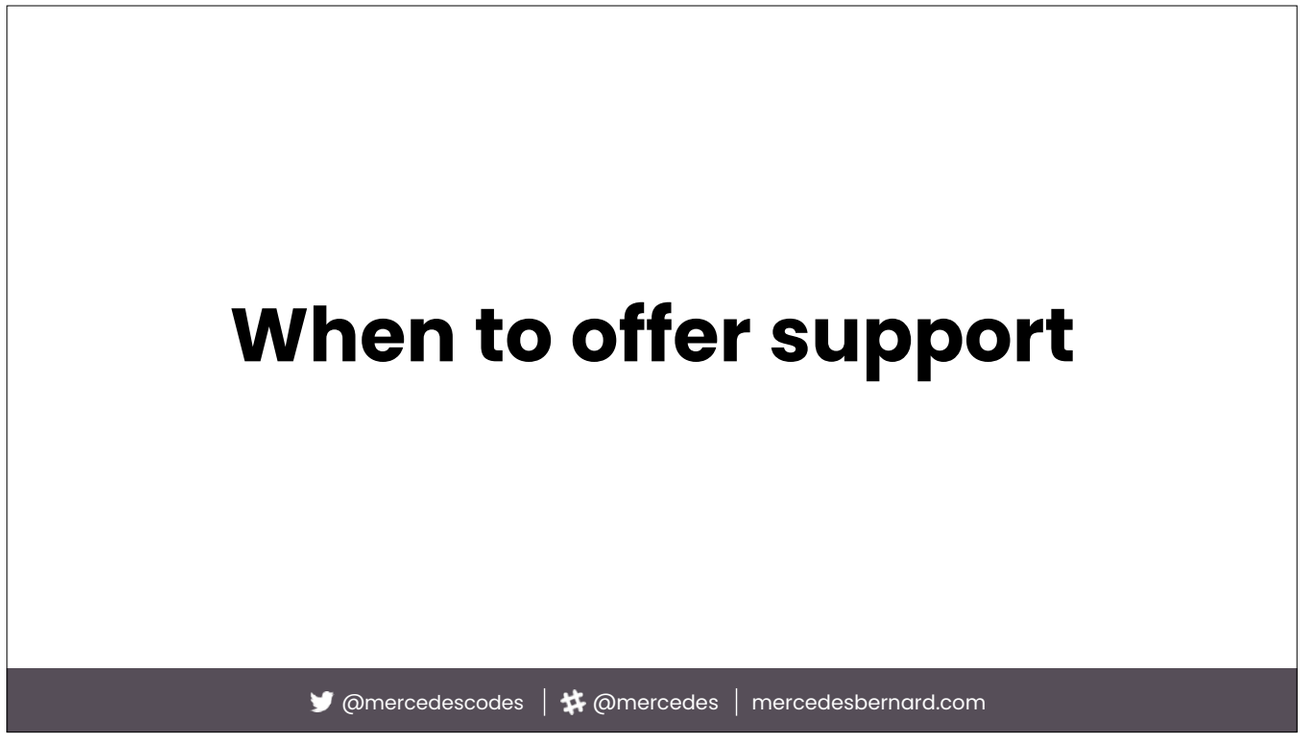
I’ve hinted at it before but my biggest learning is that delegating is hard. It can be really challenging to clearly explain what you’re looking for and to fight the urge to just do it yourself because it will go faster.

And as a lead, you need to be committed to working on this skill for your team to have as many opportunities as possible to grow.

Feedback is really important throughout this process. First and foremost, make sure that you're providing feedback to your team about what they are doing amazing. Point out their growth for them. Fight that imposter syndrome!

Also make sure to give them constructive feedback so they can continue to grow. Make it feel so normal to give and receive feedback that constructive feedback isn't scary. It can be something as simple as, don't be afraid to redirect a conversation with a client when they go on tangents like that. Some strategies I like to use are…

And also remember to solicit feedback! One piece of feedback I received during this process was to prep with one of my team members more before they facilitate a meeting. That's what they needed to feel more successful and so its something I'm more aware of now. It was super helpful feedback!



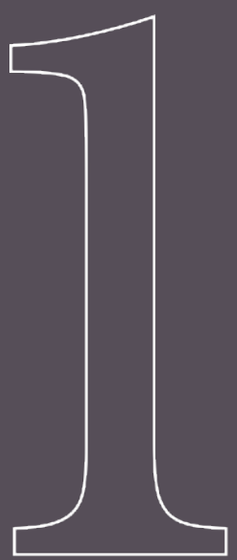
I’ve been learning when to offer support and when to take a step back. A lot of the time, its as simple as asking “How much support would you like on this?” “Would you like me to introduce the meeting and set expectations?” “Would you like to check in and make an agenda for the meeting?” “Do you need anything from me to facilitate this meeting?”

We’re going to come back to this idea of offering support in a minute, but as a lead make sure that you aren’t suffocating your team and know when to let them fly.

And finally, remember to check-in especially as your team members move through this process. It can be easy to make assumptions about the types of challenges your team is ready for but they know themselves best. Check in with them, how they’re feeling, what skills they want to work on, and incorporate that into how you iterate on your process.

How can you bring this framework into your team?

Prepare > Propose > Practice



Prepare

Identify skills you value

Identify the skills you, your team, and your company value in your senior engineers

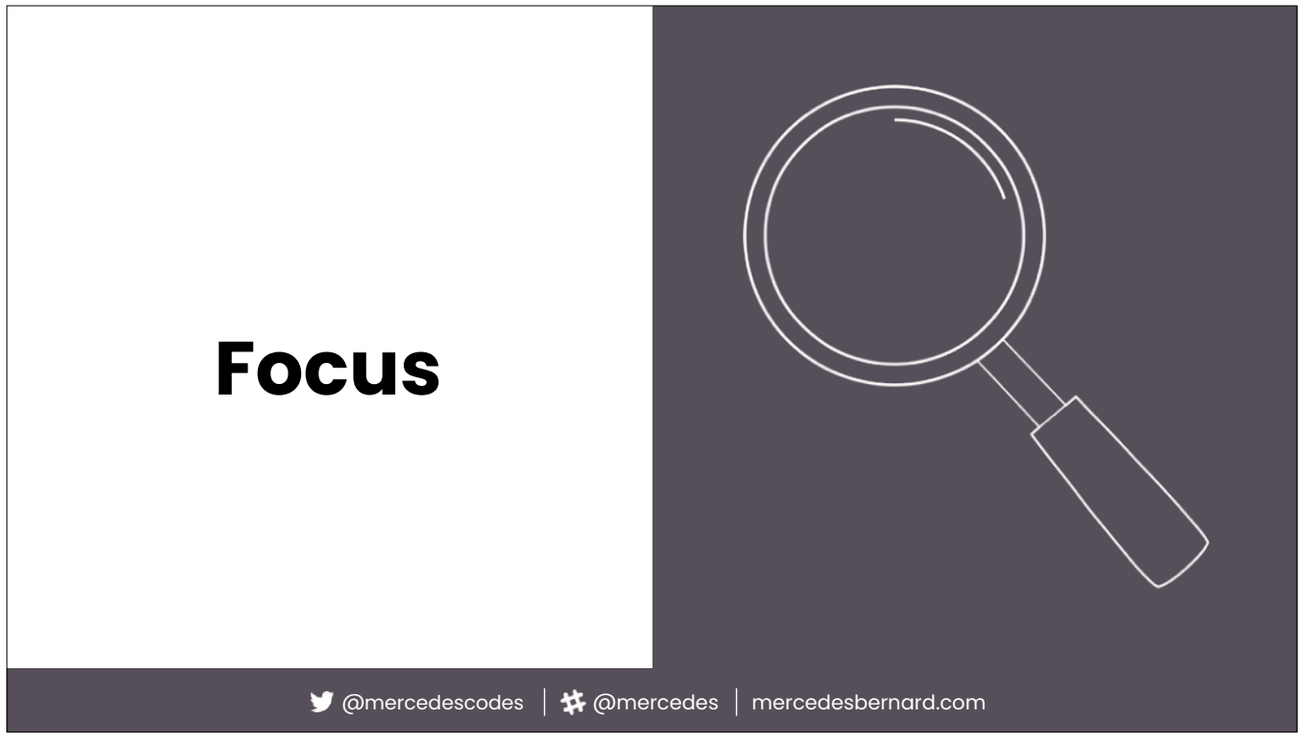
Skills

- Technical ability - new framework, data modeling, software testing, algorithms, etc.
- Technical communication
- Time management

Some ideas

Skills

- Product vision
- Mentorship
- Consulting
- Client engagement

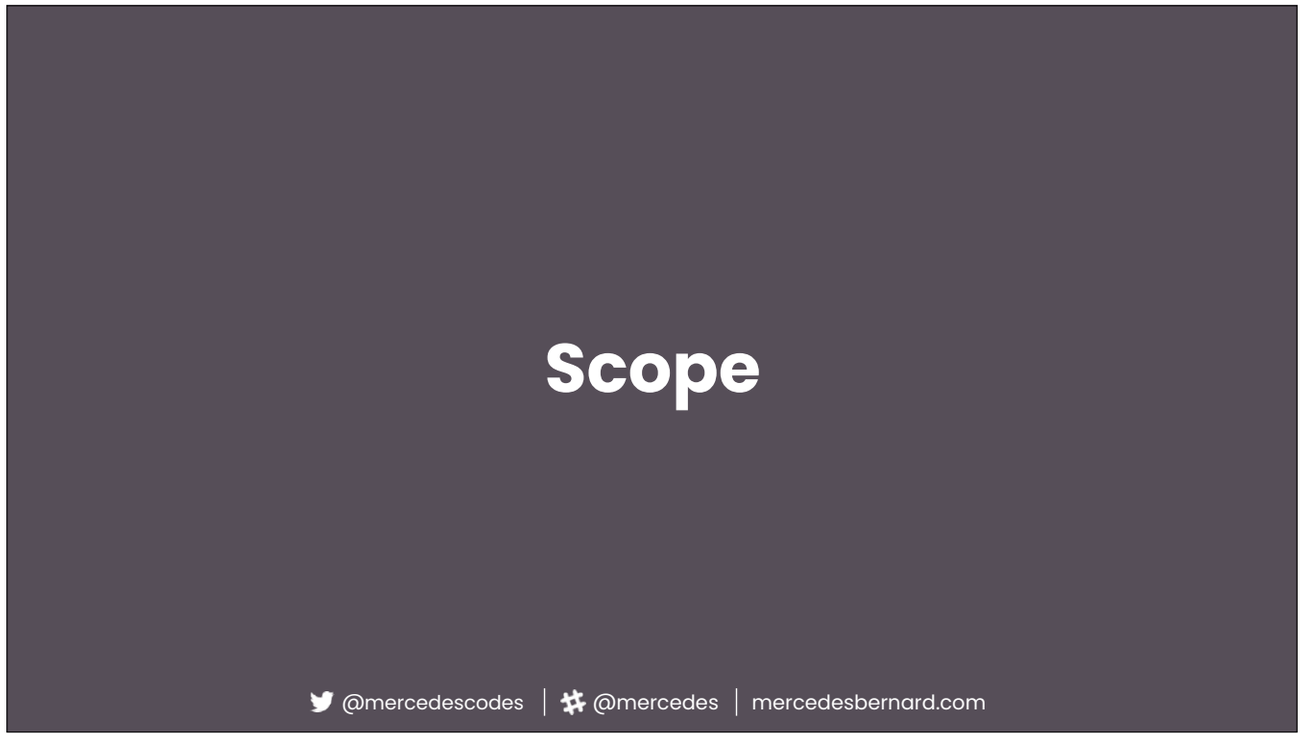


From those skills, identify which ones you want to encourage and grow on your team.

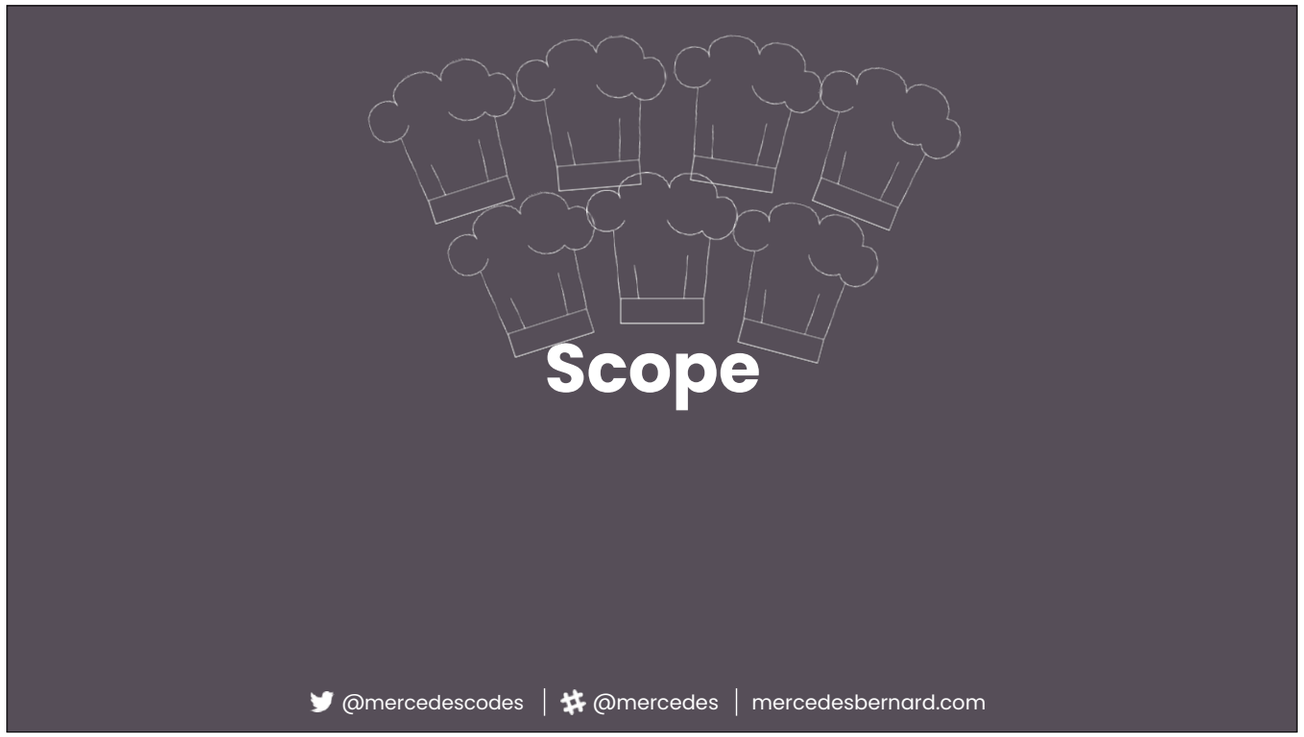
Focus

- How are you (as a lead) a silo on the team?
- What skills do you wish your team got to practice?
- What skills are needed for individuals to level up?

Some helpful questions to ask yourself might include



Next, make sure the project or problem that you’re working on large enough to accommodate shared responsibility. Will having everyone share more responsibility decrease overhead or increase it?



You want to make sure you won’t run into the "too many cooks in the kitchen” problem.

Our project is an 18 month project with many distinct application areas. There was plenty of work to go around and having everyone on the team share responsibilities would lighten my load enough to be able to offer more 1:1 coaching which was a goal of mine.

Purpose

Define the purpose and responsibilities.

You want it narrow enough to be clear, but broad enough that there are a lot of different opportunities for your team to grow.

Purpose

For all members of the team to gain consulting, client engagement, and leadership experience

Notice that for us, this was more focused on “soft” skills and less on technical expertise.

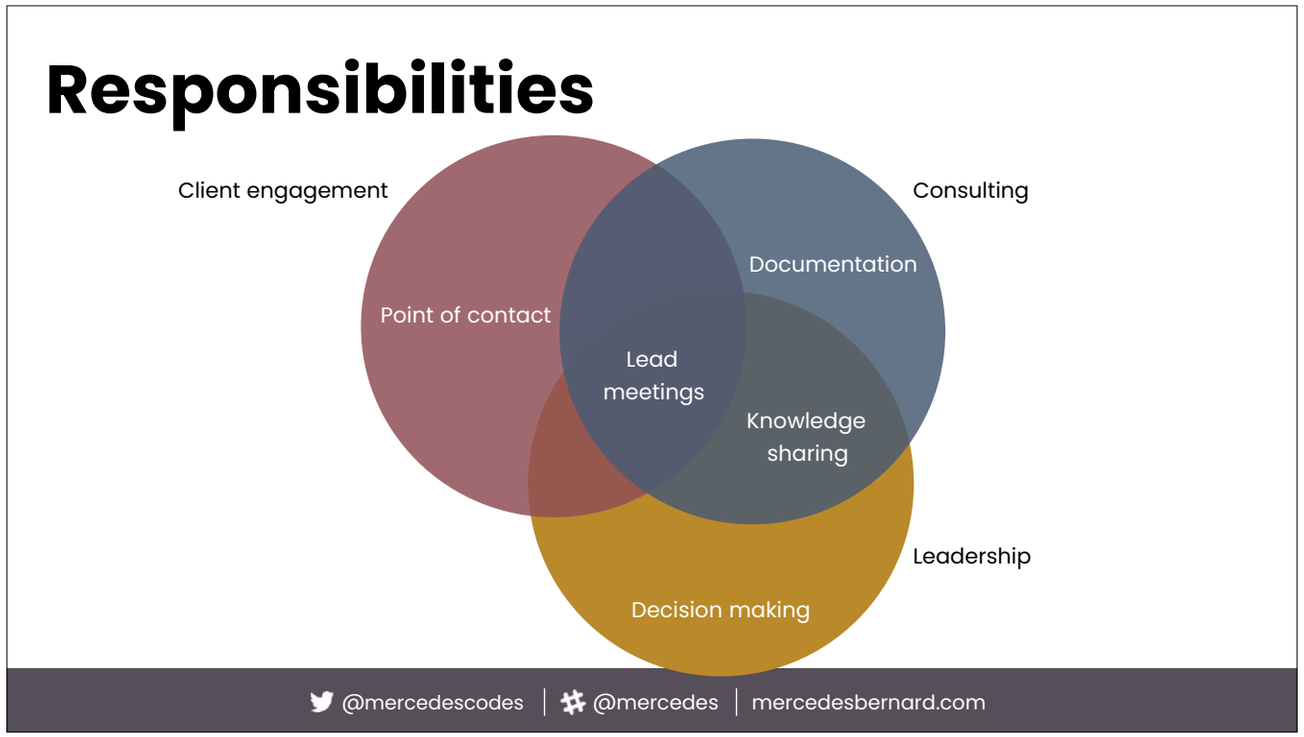
Responsibilities

From there, you want to decide on roles and responsibilities that will help achieve the purpose.

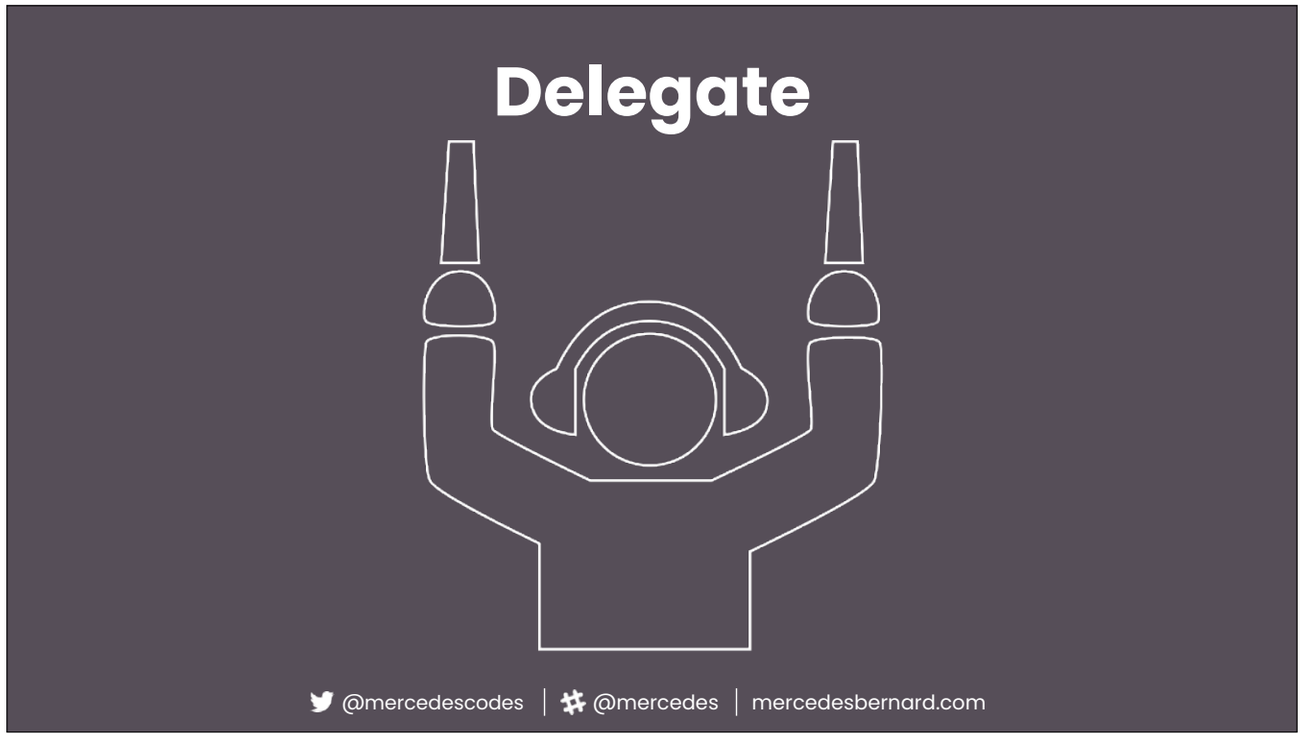
Responsibilities

- What responsibilities satisfy one or more of the skill areas you chose to focus on?
- What types of tasks would serve the purpose you identified?
- What responsibilities do you have as lead that everyone on the team could share?
- What responsibilities do you find challenging to keep up with?

I used the following questions to help me identify what those responsibilities might be. These may help you as well.



Try to identify 3-7, any less becomes really easy to ignore and any more become really difficult to focus on. It helps to write it out and it’s ok if they overlap (it might be even better). The clearer you make you responsibilities, the easier it is to track and talk about growth.



I can’t say it enough. Start mentally preparing yourself to delegate. You want to make sure that you are giving your team the space they need to practice their skills. Don’t do it for them.


2
Propose
@mercedescodes | @mercedes | mercedesbernard.com

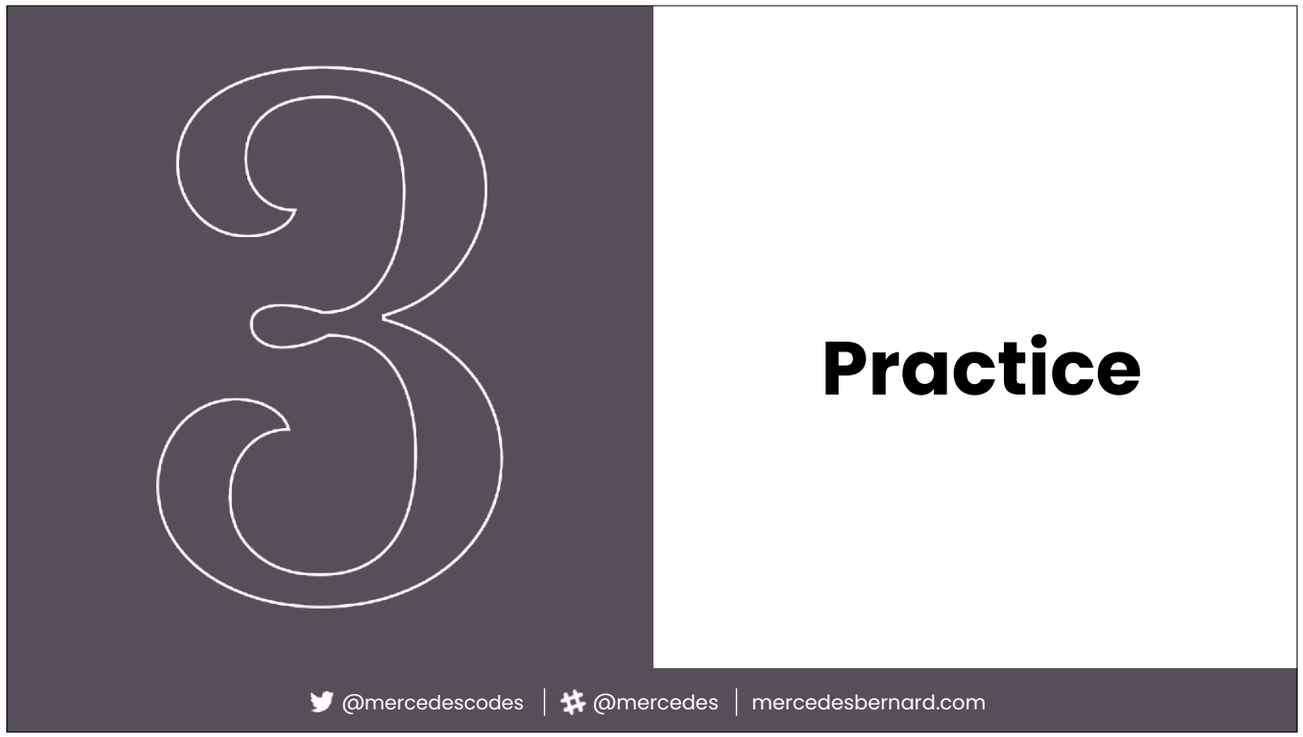
Schedule time with your team to introduce what you prepared. Don’t be afraid to have fun with it. Even though this feels like a formal framework, you can be as formal or informal as you like. I introduced the new responsibilities to my team with Wizard of Oz themed slides.

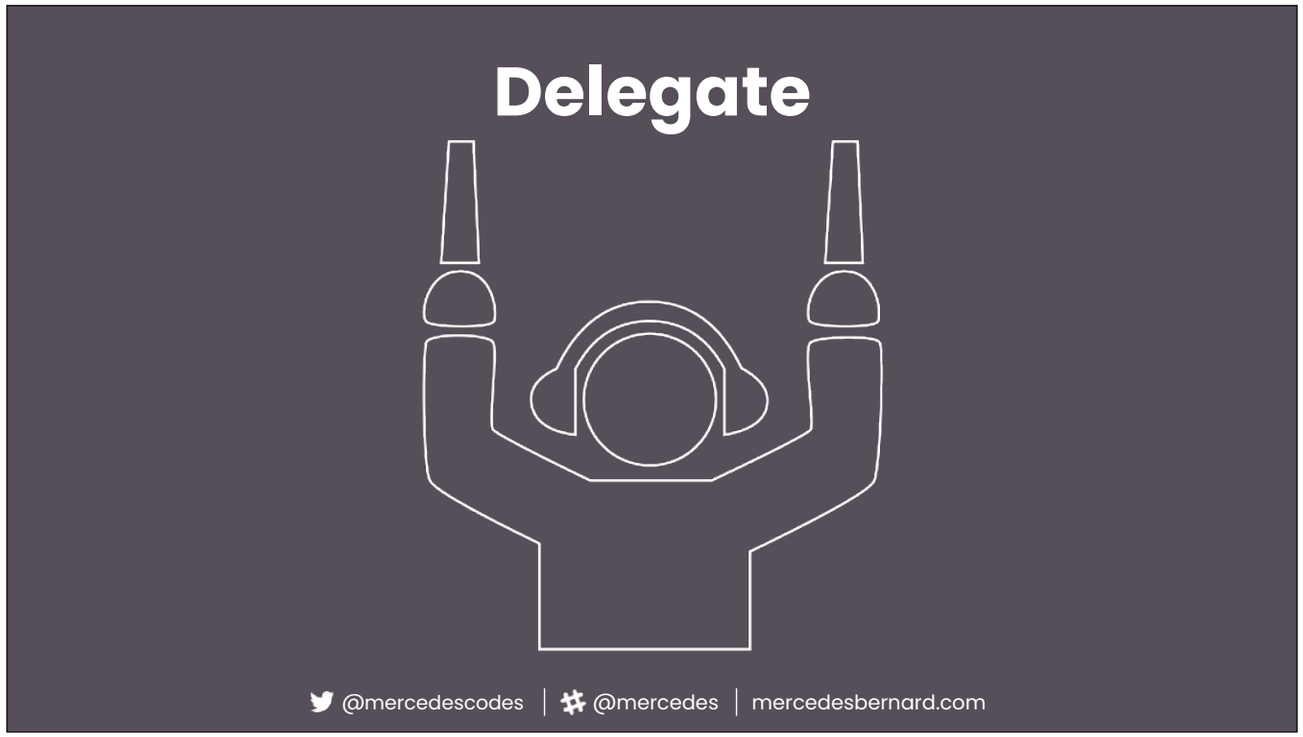
Solicit feedback

Solicit feedback

- Are there skills your team would like to practice that aren't covered?
- Does the team feel comfortable but challenged by the responsibilities?
- Is there anything your team wants to add or clarify?
- Does your team feel that this implementation will be sustainable?

The following questions are a good place to start


3
Practice
@mercedescodes | @mercedes | mercedesbernard.com



Now it’s time to follow through. Support your team, but let them take it from here.

Offer Support

There are many ways to offer support and a lot of times they are small ways of providing encouragement and feedback.

Offer Support

Regular 1:1s

Offer Support

Reading an email

Offer Support

Prep before a meeting

Offer Support

Pairing on story writing

Offer Support

Pairing on data modeling/architecture

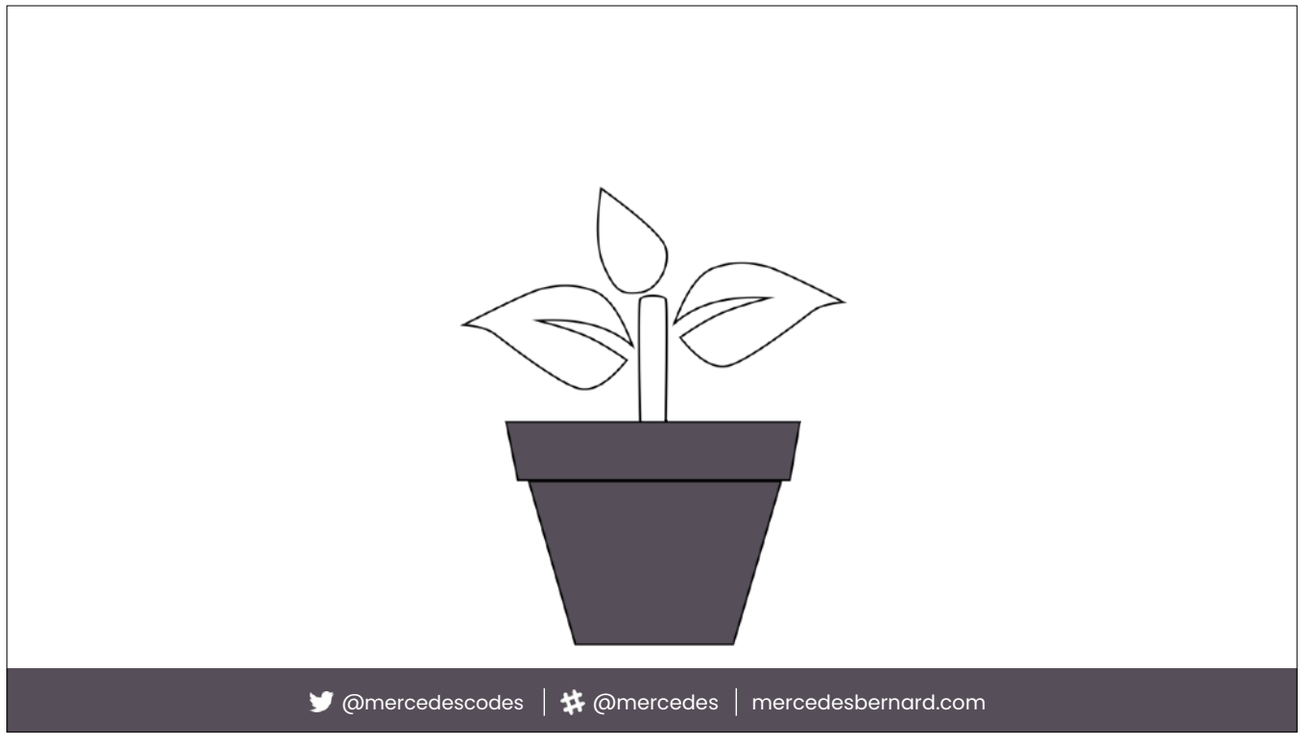
Offer Support

Calendar invites for those activities

Offer Support

Reminders about deadlines

Offer Individualized Support



Our industry is missing out on so much talent because we don’t do enough to support the growth of those early in their career. We all need to do our part to empower our early career developers and grow them into the amazing seniors and leads we all know they can be.

Thank You

@mercedescodes | @mercedes | mercedesbernard.com

mercedesbernard.com/speaking

@mercedescodes | @mercedes | mercedesbernard.com