Congrats on becoming an engineer!

Let’s start off strong 💪

I’m Alex

- My goal: **Grow you past the junior level at light speed** 🏃
- Previously a Tech Lead at Course Hero, Meta, and Robinhood
- Prior to Taro, making $750k/year as a top TL leading 15+ engineers
- Coached dozens of engineers to ⚡ senior promotions at top companies

Junior 🤓

Meta

E3

$192,702

Alex's Mentorship 🪴

1 year promotion

10+ Engineers

Mid-Level 🙂

Meta

E4

$306,684

*A few mentees got promoted in just 6 months!

What's Junior?

- You start off here (usually)
    - 0 - 4 YOE, small band
- Usually a 4-year Computer Science degree is required
    - Might need to do an apprenticeship without
- Junior jobs can be competitive for those with actual 0 YOE

L3
SWE II

E3

SDE I
L4

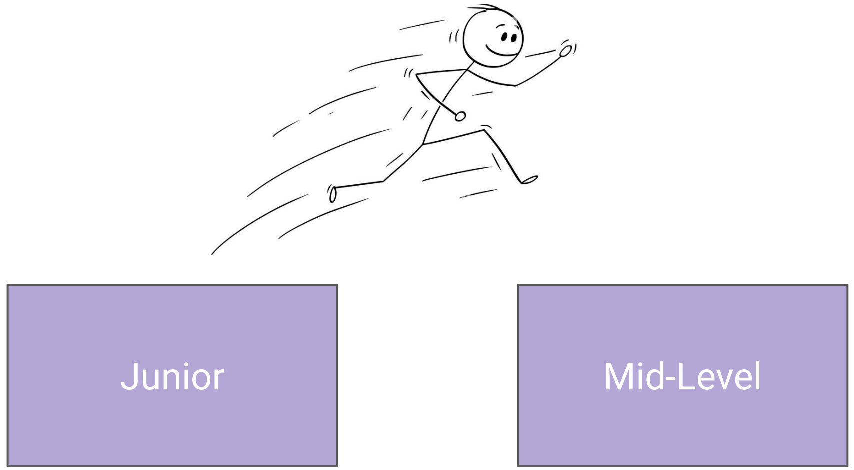
What's Mid-Level?

- Most engineers take *2-4 years* to get this promotion
- The median software engineer is in this band
    - Very wide: 2 - 10 YOE
- 2+ YOE to get the interviews
- Is often a terminal level
    - Google L4
    - Amazon SDE 2

L4
SWE III

E4

SDE II
L5



Every engineer *eventually* figures out junior -> mid-level.

The real question is...

How fast? 🏃 🤔

This course will get you there **fast** 🚀

Objectives

- 💡 Truly understand the **difference** between a junior engineer and a mid-level engineer
- 🛠️ Learn to **code** like a mid-level engineer
- 💪 Know what it means to be truly **independent**
- 🧠 Have the right **mindset** to act as a mid-level
- ✅ Define a clear **roadmap** to mid-level

The Difference Between Junior & Mid-Level

L3 SWE II	Life just happens to them. Very reactive and not in control. Get tasks, do tasks
L4 SWE III	Takes control and owns their destiny on execution. Acts more proactively

L3

SWE II

Able to get tasks in on-time but not much else. Needs hand-holding. People worry about their ability to deliver

L4

SWE III

Proven executor. Gets tasks in on time with high quality. Minimal hand-holding, people don’t worry about them

L3

SWE II

Works on low to medium complexity tasks that are **1 month or less**

L4

SWE III

Works on medium to large complexity tasks (project slices) spanning **1-3 months**

L3 SWE II	Not an expert on anything
L4 SWE III	Respected expert of their codebase. People happily come to them with questions, trusting the answers

L3

SWE II

1 engineer
(themself)

L4

SWE III

1 - 3 engineers

It’s On You

Which Junior Engineer Gets More Credit?

- Has a **great** product manager
- Gets great projects that add massive user value and **hit goals**
- Delivers on-time and with high code quality

- Has a **mediocre** product manager
- Gets projects that usually miss and **don't contribute to goals**
- Delivers on-time and with high code quality

*They get the **same** amount of credit!*

L3/L4 are generally not held responsible for impact. 🙏

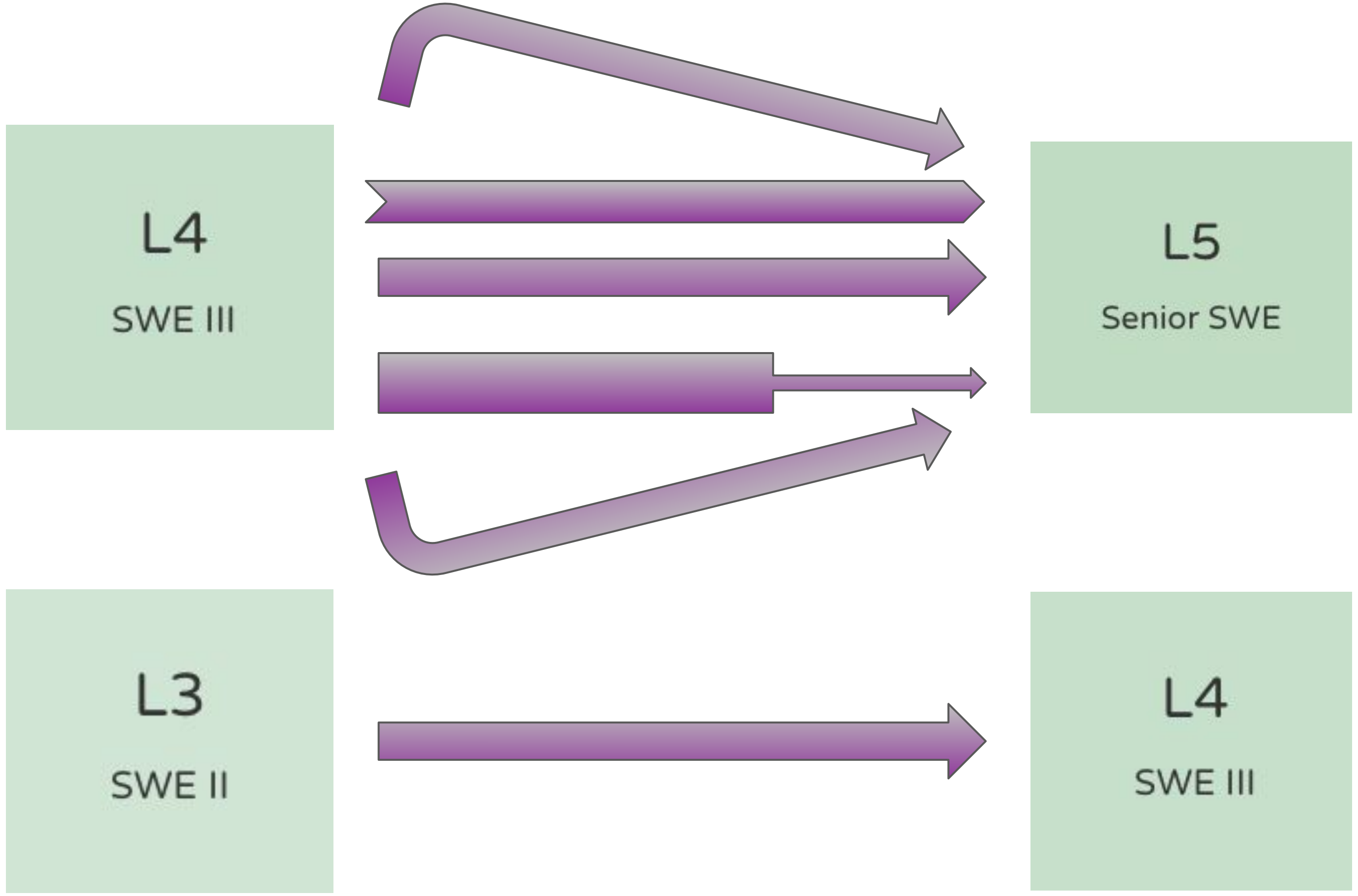
It's All About *Execution*

1️⃣ You don't need to find a team with the "best scope"

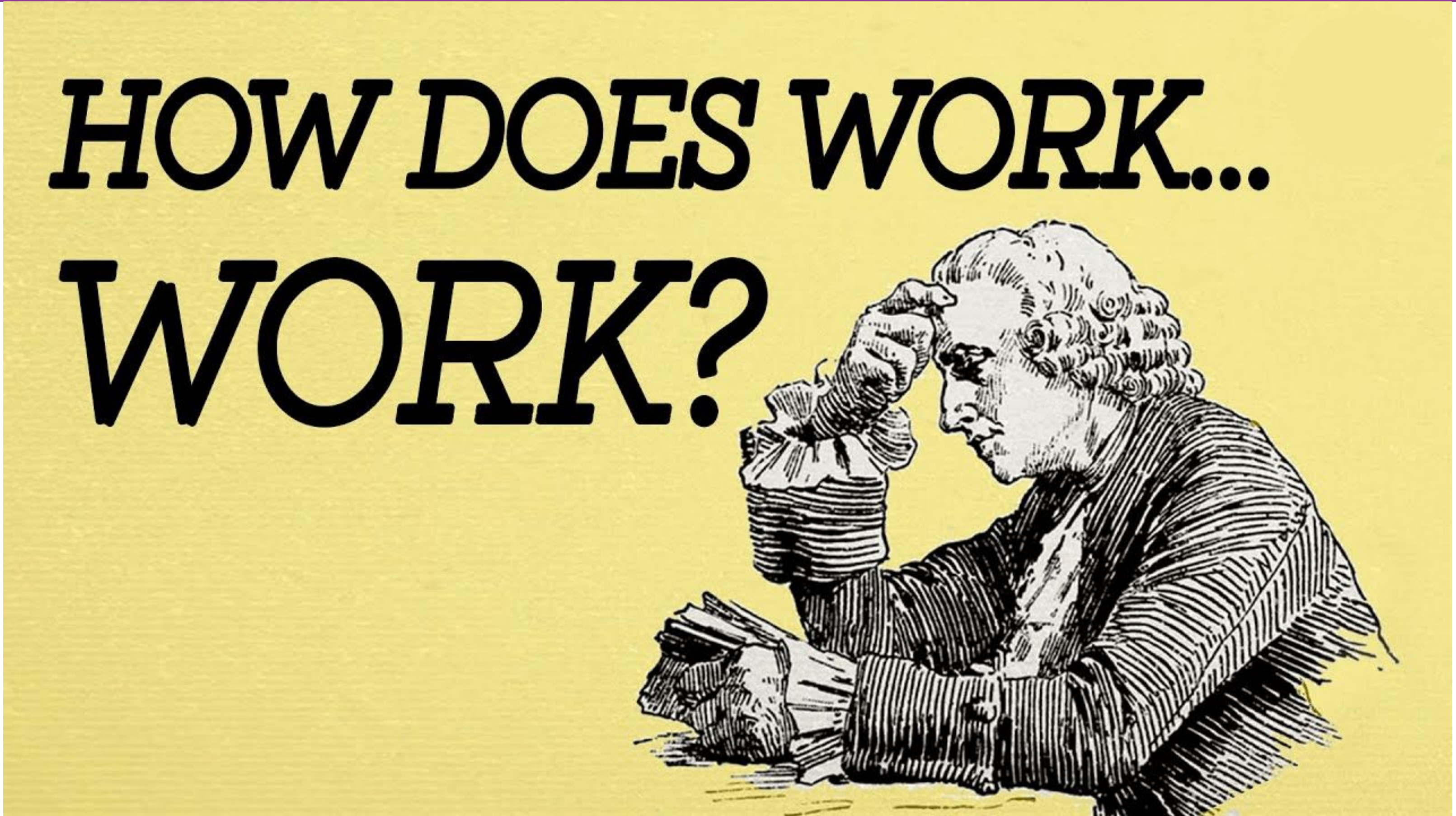
2️⃣ You don't need to work on the "best projects"

3️⃣ Sharpen individuals skills + synergy with the team

Nail Your Promotion As A Software Engineer


L4
SWE III
L5
Senior SWE
L3
SWE II
L4
SWE III

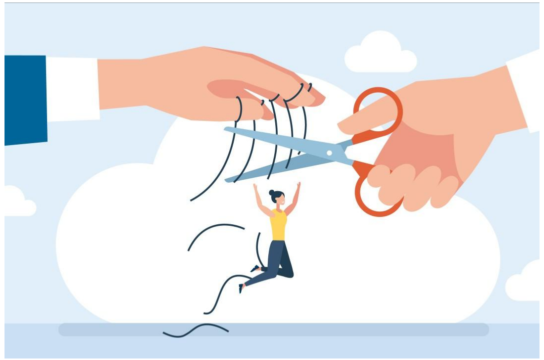
Structuring Your Growth



The 3 Areas Of Growth For L3 -> L4

The Code 🧑‍💻

Independence 💪



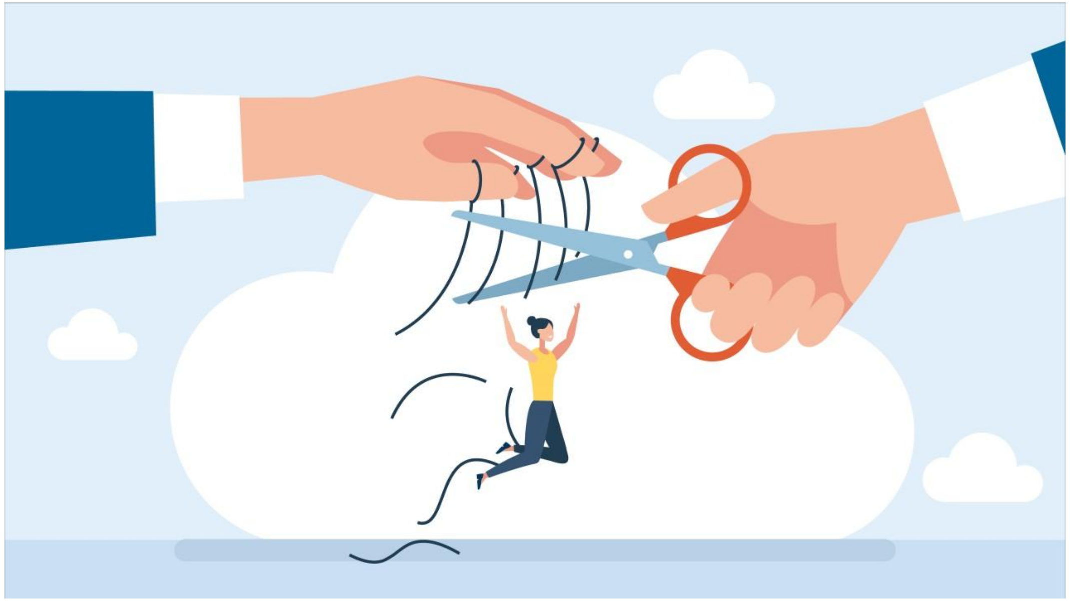
Collaboration 🤝

The Code 🧑‍💻

Definition: Every interaction you have with code (50% - 70% of L3 -> L4)

Independence 💪

Definition: Your ability to figure things out on your own (i.e. without hand-holding)



Collaboration 🤝

Definition:
Harmonizing with the broader team and adding value to others

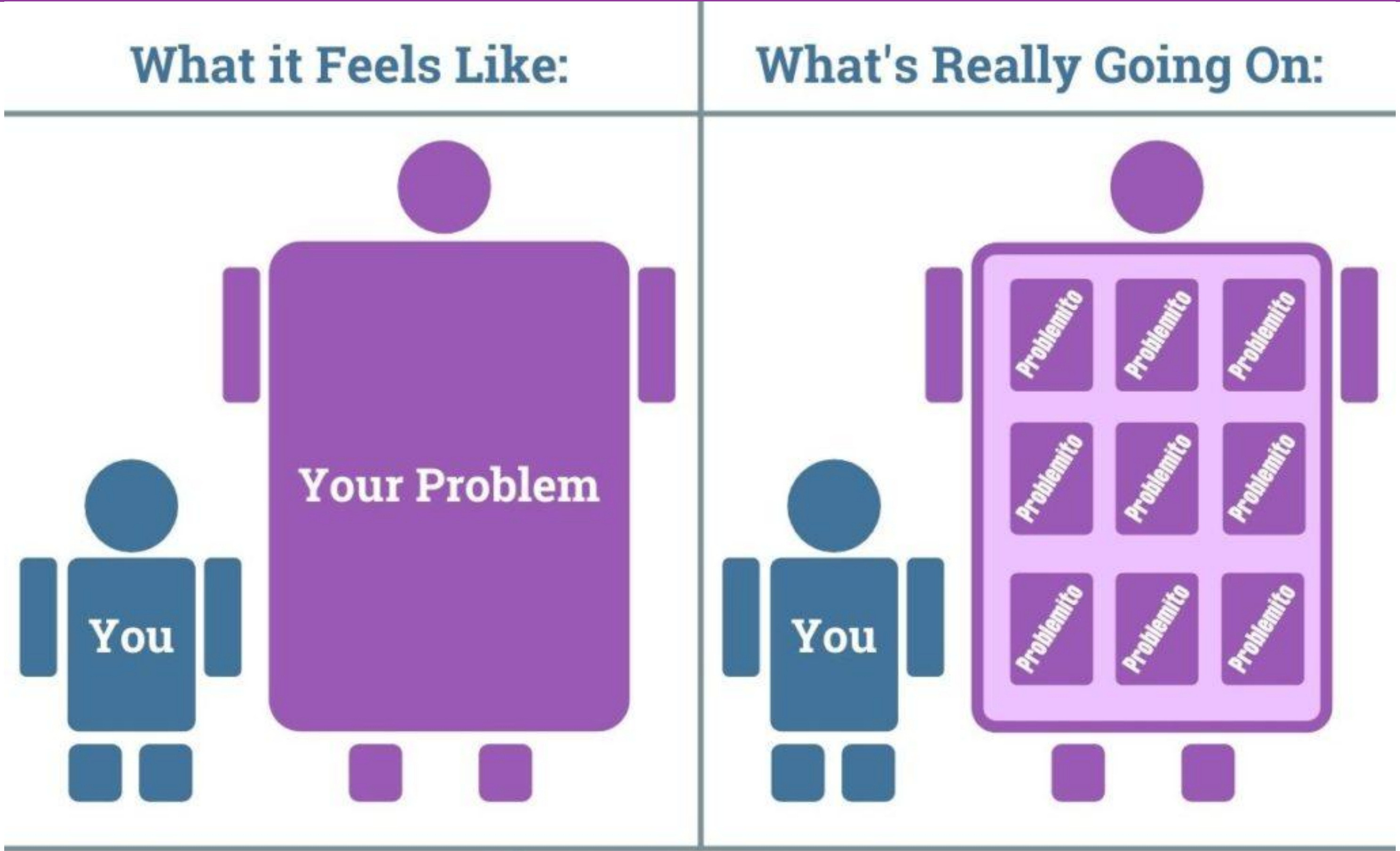
Pictured: A junior engineer growing to a mid-level engineer (totally real and 100% accurate)

This **Is** A Checklist

- You *cannot* grow past junior while missing several of the items in this course
  - Revisit the course and see if you're meeting the L4 bar in each lesson
- L3 -> L4 is pure foundation
  - Table-stakes to be a solid engineer

The Code

Decomposition



Break It Down

- Any scary problem can be broken down into smaller sub-problems
- >1 week = Decompose
- Very important for junior engineers as most problems are scary
- Critical for *both* code velocity and quality

Decomposition: Junior vs. Mid-Level

L3

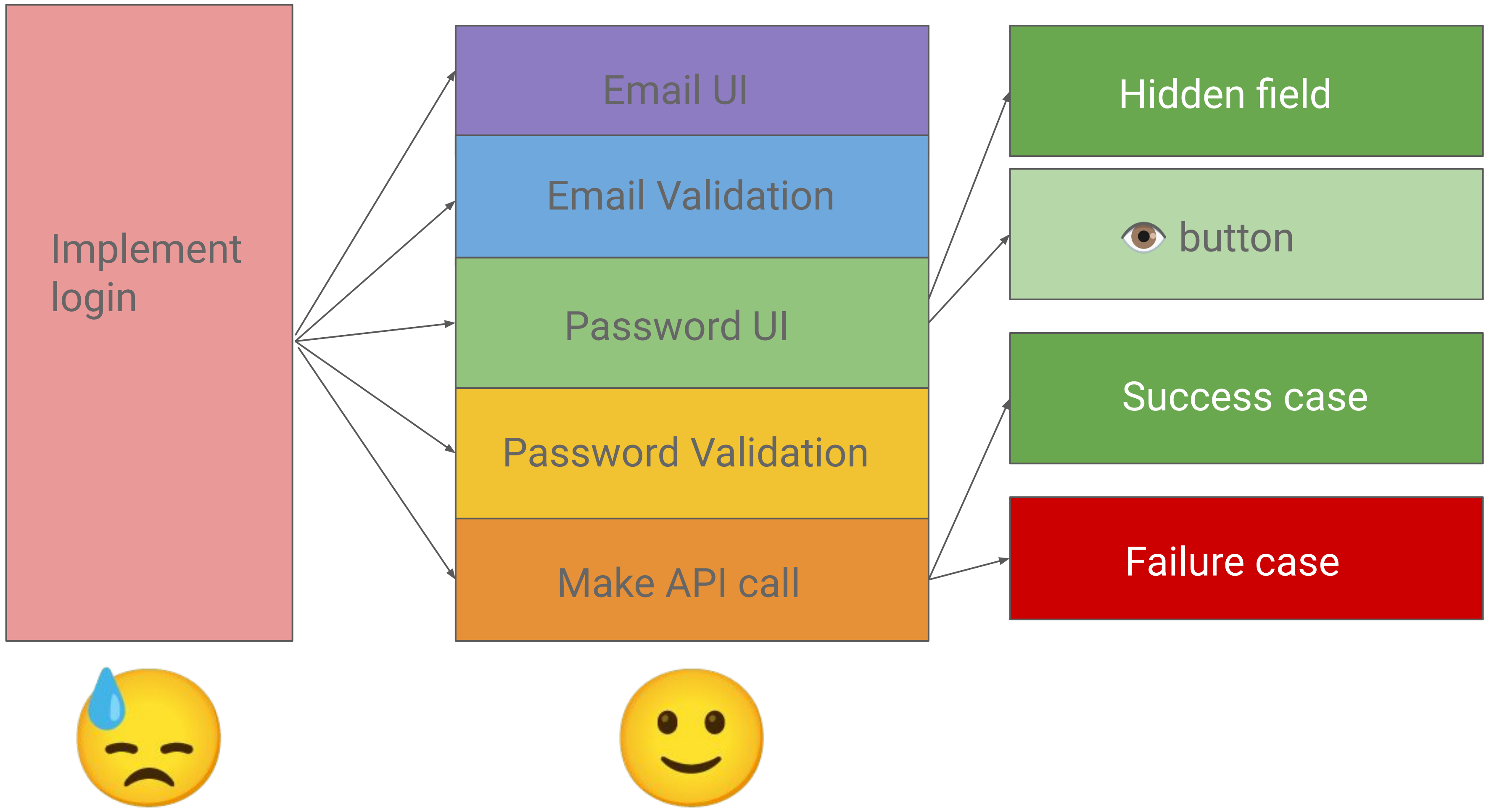
SWE II

- No decomposition of the problem, runs headfirst into the code
- Submits giant PRs
- Coding lone wolf

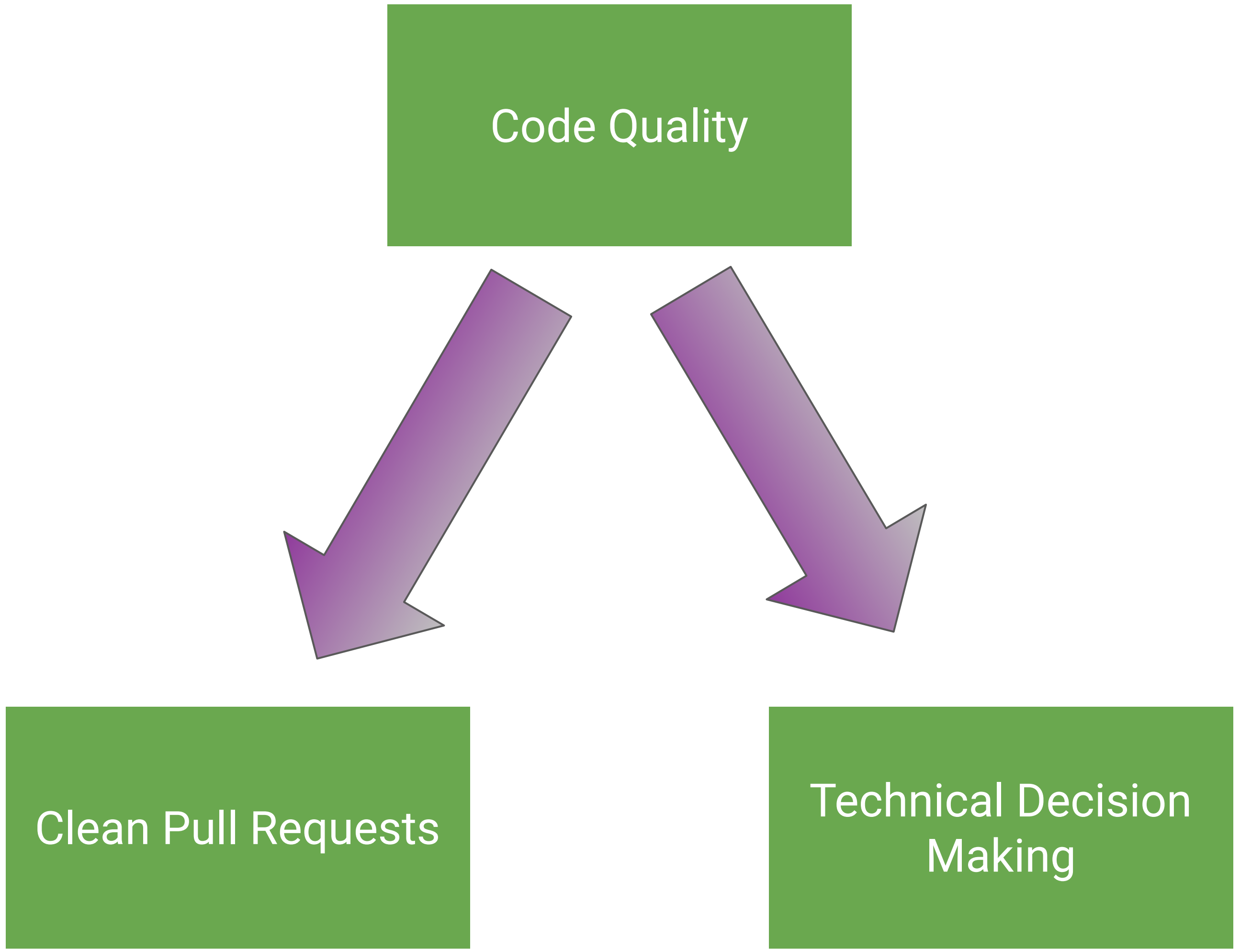
L4

SWE III

- Decomposes the task into sub-problems, creates a plan
- Submits focused PRs
- Gets buy-in when necessary


Implement login
Email UI
Email Validation
Password UI
Password Validation
Make API call
Hidden field
👁 button
Success case
Failure case

Code Quality


Code Quality
Clean Pull Requests
Technical Decision Making

Clean Pull Requests (PRs): Junior vs. Mid-Level

L3

SWE II

- Pull requests aren’t clean at all (just a raw vessel for code)
- They only care about the code itself
- Hard to review

L4

SWE III

- Pull requests are tidy and organized
- Realizes the “dressing” around the code matters greatly
- Joy to review

Clean Pull Requests 🧼

1️⃣ *Follows "One Diff, One Thesis"*

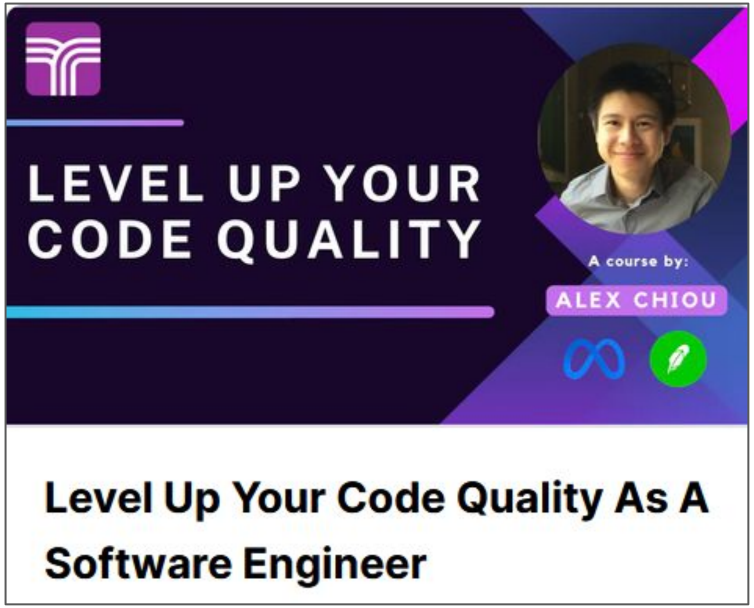
2️⃣ Descriptive "Plain English" Title

3️⃣ Detailed context section

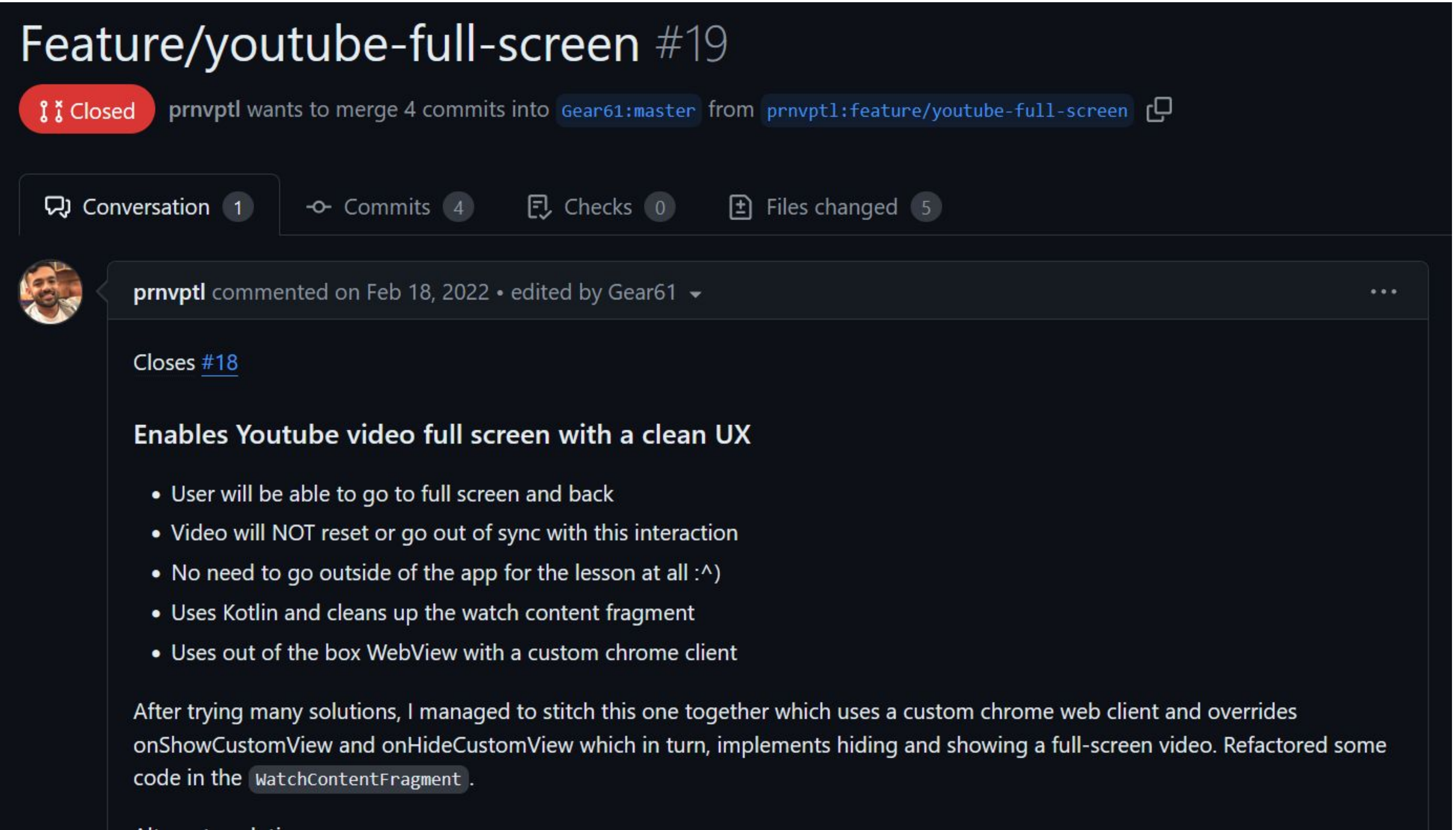
4️⃣ *Thorough test plan*

5️⃣ PR self-comments

6️⃣ Proper commit names


LEVEL UP YOUR CODE QUALITY
A course by:
ALEX CHIOU
Level Up Your Code Quality As A Software Engineer

Sneak Peek: What A Clean PR Looks Like 🫣



Feature/youtube-full-screen #19

Closed prnvptl wants to merge 4 commits into `Gear61:master` from `prnvptl:feature/youtube-full-screen`

prnvptl commented on Feb 18, 2022 • edited by Gear61

Closes #18

Enables Youtube video full screen with a clean UX

- User will be able to go to full screen and back
- Video will NOT reset or go out of sync with this interaction
- No need to go outside of the app for the lesson at all :^)
- Uses Kotlin and cleans up the watch content fragment
- Uses out of the box WebView with a custom chrome client

After trying many solutions, I managed to stitch this one together which uses a custom chrome web client and overrides onShowCustomView and onHideCustomView which in turn, implements hiding and showing a full-screen video. Refactored some code in the `WatchContentFragment`.

Submit clean pull requests (your teammates will love you for it).

Technical Decision Making: Junior vs. Mid-Level

L3

SWE II

- Often chooses a suboptimal approach
- Code is messy
- Product performance isn’t sharp
- The code merely works

L4

SWE III

- Often chooses the optimal approach
- Code is polished
- Sharp performance
- The code not only works, but it works well

L3

SWE II

</>

"Here's my pull request."

"Uhh, you'll need to rewrite half of this. The code is extremely hard to read and fragile."

Teammate

L4

SWE III

“Here’s my technical plan to accomplish this meaty ticket. I attached as much pseudo-code as I could.”

📄

🙂

Teammate

“This looks great! Just make sure to handle the case where the user doesn’t have internet access too.”

The *communication* around your technical decisions also matters. 🗣️

How To Make Better Technical Decisions 🤔

1️⃣Decompose your tasks

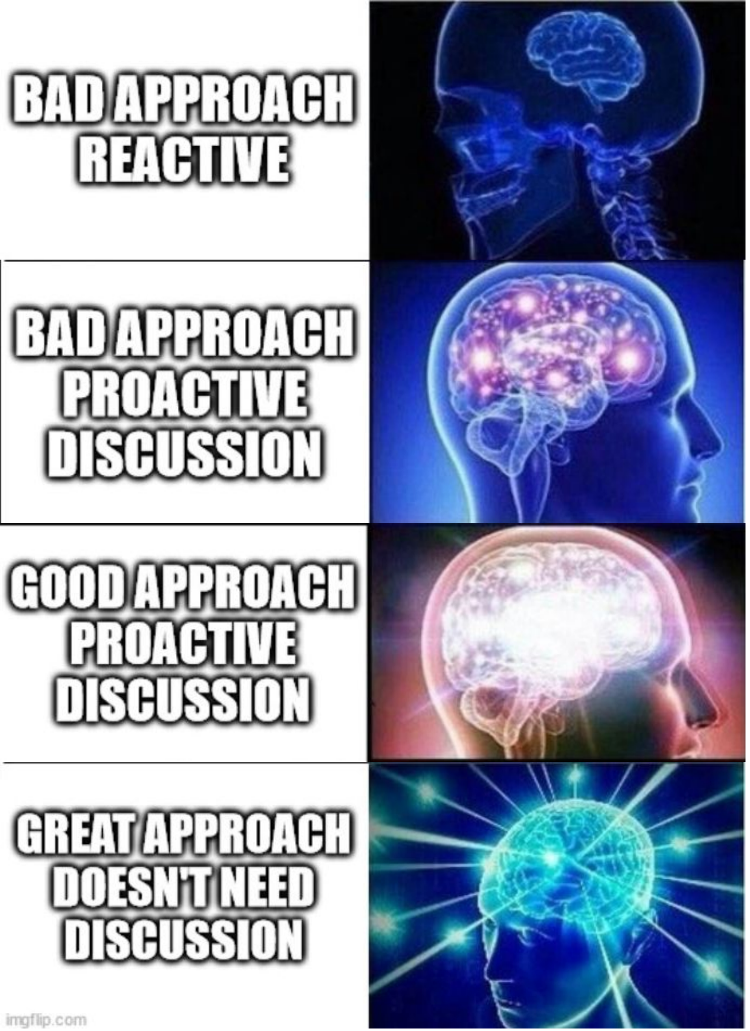
2️⃣Communicate your plan proactively

3️⃣Learn from your mistakes (retrospect)

4️⃣See how other pull requests are rejected

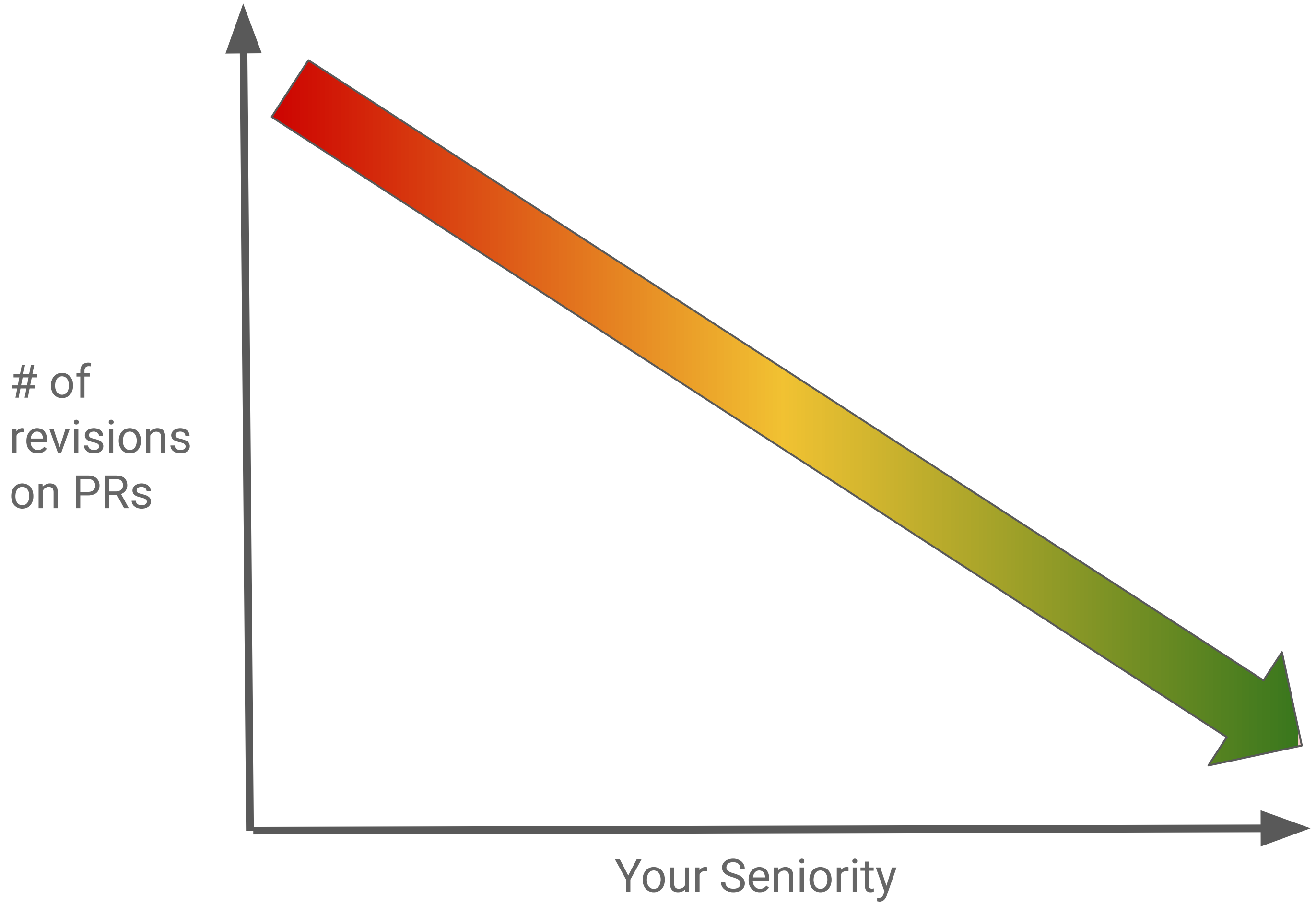
5️⃣Listen to system design meetings

6️⃣Think through edge cases


BAD APPROACH REACTIVE
BAD APPROACH PROACTIVE DISCUSSION
GOOD APPROACH PROACTIVE DISCUSSION
GREAT APPROACH DOESN'T NEED DISCUSSION
imgflip.com

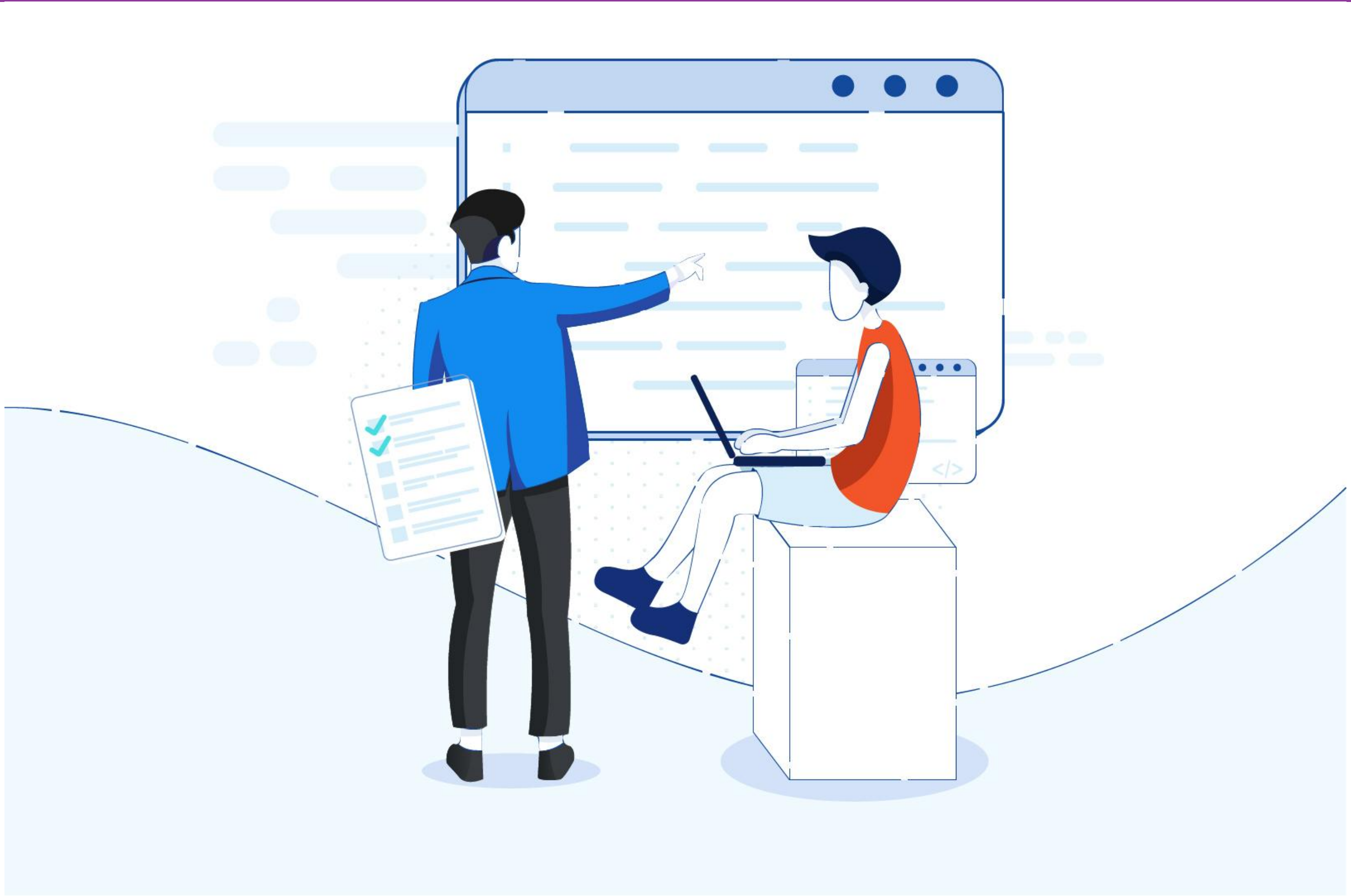
Technical Decision Making Progression (Task Level) 🪜

L3 -> Strong L3 -> L4 -> Strong L4


# of
revisions
on PRs
Your Seniority

Become an expert craftsperson.

Code Review



L3

SWE II

125 commits landed

60 commits reviewed

L4

SWE III

150 commits landed

175 commits reviewed

Mid-level engineers use their expertise to level up other's code 💡

“The indentation of this method is off.”

L3

SWE II

“We could move this code out of the ViewController and into a helper class, which we connect back via a delegate.”

L4

SWE III

Debugging

Debugging: Junior vs. Mid-Level

L3

SWE II

- Can barely fix bugs
- Bugs need to be within code they've worked on before
- Tackling low complexity issues

L4

SWE III

- Fixes bugs regularly, especially in their area
- Can adapt to adjacent codebases
- Tackling medium/high complexity issues

What Makes A Bug Complex? 🪲

1️⃣ Performance related, not feature related

2️⃣ Not 100% repro

3️⃣ No repro steps

4️⃣ Outside of your codebase

5️⃣ Cross-domain

Become A Go To Person

L3

SWE II

“I work on this codebase.”

</>

L4

SWE III

"This codebase is my baby, and I will give my all taking care of it."

</>

Tips For Becoming A Go-To Person 🧙‍♂️

1️⃣ Deliver on-time and with clear communication

2️⃣ Deliver with high quality (smooth PRs, few bugs)

3️⃣ Hold down the fort (fix bugs, put out 🔥s)

4️⃣ Answer questions in your area (be a kind steward)

5️⃣ Participate in system design meetings

Become The Go-To Expert As A Software Engineer

Dive deeper into becoming a go-to person with our course! 📚

Think like an owner.

Independence

Be Vulnerable

Why Vulnerability Is Important
Greater strength
Stronger relationships
Improved self-acceptance

True Strength

- True strength is being open with your weaknesses instead of hiding them
- Awareness and acceptance are the first steps
- *You become independent by being very dependent for a short period of time*

Independence Mindset: Junior vs. Mid-Level

L3

SWE II

- Doesn’t ask for nearly enough help
- Believes that needing support is a sign of weakness

L4

SWE III

- Asks for help when they need it
- Realizes that connecting the dots is what it means to be a software engineer

Don’t kick the can down the road when it comes to your weaknesses 🦵🥫

Embrace them and share them. You can’t become a good engineer on your own 👐

Ask Good Questions

Asking Questions: Junior vs. Mid-Level

L3

SWE II

- Questions are low-quality, placing a lot of burden on helpers
- Questions are purely looking for help
- Language is very raw and basic

L4

SWE III

- Questions are high-quality, absorbing burden themselves
- Questions demonstrate expertise
- Language is advanced and elegant

L3

SWE II

"I am trying to load an image in my Android app, and it shows up blank. Why?"

"Where is the image coming from?"

"Are you using a library?"

"Did you declare the INTERNET permission?"

✗

L4

SWE III

"I am trying to load an image from this URL using Picasso: tinyurl.com/9xu67

My app has the INTERNET permission, and my emulator can connect to wifi. My ImageView has the proper bounds.

I already tried X, Y, and Z from StackOverflow.

<Links to additional resources with more context>

"I've seen this before! Your image URL is a redirect, so just add this Picasso extension module."

✓

Ask Great Questions That Get Great Answers Quickly

Become a master question asker with our course! 📚

(2nd most important piece after code quality)

Disambiguation

Disambiguation: Junior vs. Mid-Level

L3

SWE II

- Needs tasks to be +90% defined
- Might not realize a task isn’t fully defined
- Often needs to ask an EM/TL/PM to clarify

L4

SWE III

- Can figure things out as long as the task is +50% defined
- Can have mature conversations with stakeholders to clarify

L3

SWE II

“The designs don’t account for the case where the user doesn’t have internet access. What should we show here?”

Designer

L4

SWE III

“The designs seem to be missing the loading state, empty state, and error state. Based on other app screens, I think we should do something like ABC. What do you think?”

Designer

Collaboration

Be A Feedback Sponge

FEEDBACK
Is a Gift
Especially
WHEN IT HURTS

Fresh Junior Engineer

"I went to a Top 25 CS school and have 500+ completed LeetCode problems. I know my stuff!"

Teammate

"Oh you sweet summer child... You'll find out how clueless you are someday..."

Intern from Princeton joins my team at Instagram

Clearly pretty smart, starts writing code quickly

Gets feedback that their PRs are large/messy

Doesn’t really incorporate the feedback (???)

Doesn’t get a return offer 👎

Handling Feedback: Junior vs. Mid-Level

L3

SWE II

- Often defensive with feedback
- Drops feedback (intentional + unintentional)
- Doesn’t ask for feedback

L4

SWE III

- Gracefully accepts feedback
- Incorporates feedback efficiently
- Proactively probes for feedback

Feedback is a gift.

1 on 1 Meetings

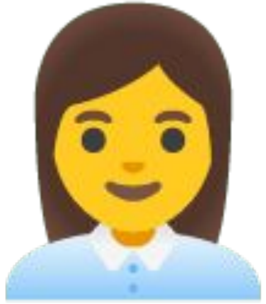
During The Meeting

L3

SWE II

"What do you want to talk about today?"

"Oh, uh... So I've been working on this task, and here's the progress I've made..."



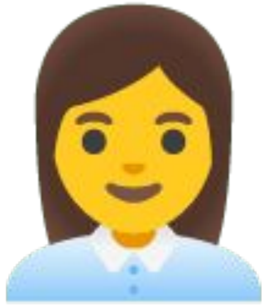
Manager

Before The Meeting

L4

SWE III

"Excited to talk to you tomorrow! I put some great topics for us to talk about in our meeting notes, ordered by priority."



Manager

"Awesome! Let's make great use of the time."

1 on 1 Meetings: Junior vs. Mid-Level

L3

SWE II

- Goes into the meeting with no plan, often doesn’t take notes
- Is the *passenger*, not the driver
- Tactical discussion

L4

SWE III

- Sets the agenda, approaches every 1 on 1 with a clear goal
- Is the *driver*, not the passenger
- Discusses deeper topics

Participating In Bigger Meetings

L3

SWE II

L4

SWE III

Tips For Speaking Up In Meetings 🙋‍♀️

1️⃣ *Don’t feel pressured to say something super insightful*

2️⃣ Do “homework” prior to the meeting (read through agenda/links)

3️⃣ Raise your hand (if you’re afraid of interrupting)

4️⃣ Reinforce something someone else said

5️⃣ Ask questions

L3

SWE II

<A bunch of stuff you sort of understand>

Teammate

“Let me say this back to you (and please correct me if I'm wrong): We can't use this external library as it's filled with security flaws?”

Case 1: You’re Right ✅

L3

SWE II

“That’s correct! Great summary.”

Teammate

- You look smart
- Clarification is great for alignment
- You show that you’re paying attention

Case 2: You’re Wrong ❌

L3

SWE II

- You learned something 💡
- Clarification is great for alignment
- You show that you’re paying attention

“Almost. We vetted the library and couldn’t find any vulnerabilities, but the company behind it has a history of breaches.”

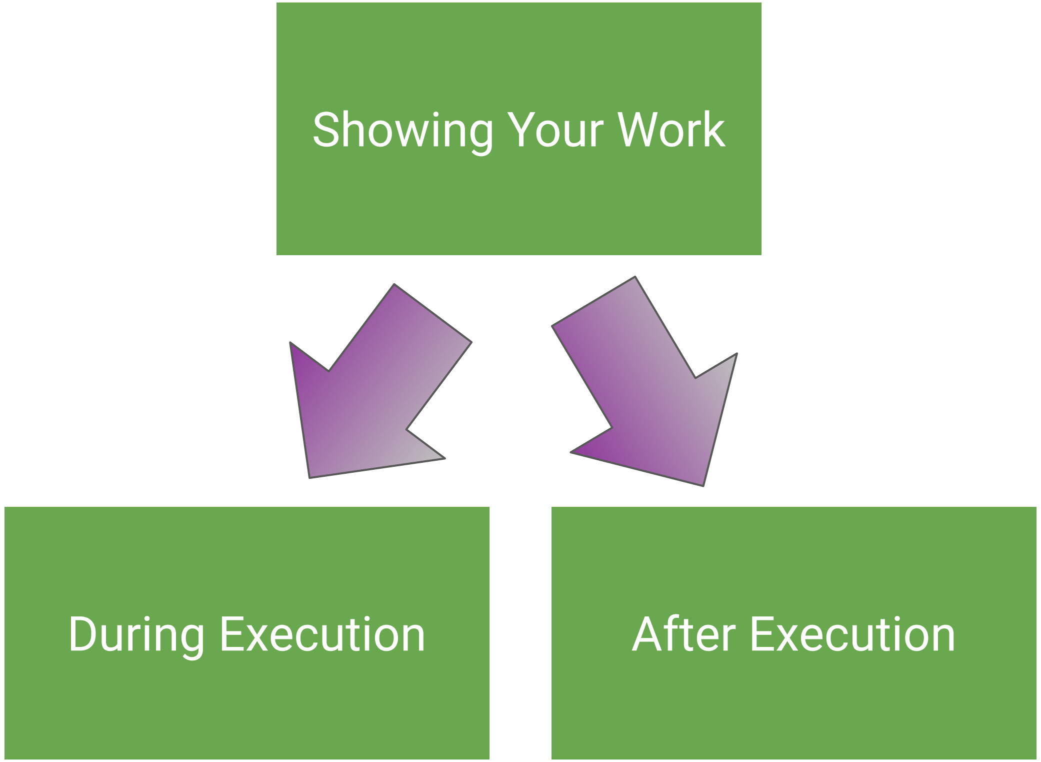
“Ah, got it. Thanks for clarifying!”

🙂

Teammate

Show Your Work

THE PAYOFFS AND HOW-TO'S OF WORKING OUT LOUD


Showing Your Work
During Execution
After Execution

Showing Your Work: Junior vs. Mid-Level

L3

SWE II

- Gets tasks, doesn’t surface for air until the due date
- After the code is landed, they’re done

L4

SWE III

- Gets tasks and shares clear, regular updates
- Anchors against milestones
- After the code is landed, they announce it (often with a slick demo)

Progress Updates: Junior vs. Mid-Level

"I'm still working on the task."

"I'm making good progress on the task. It's split into 3 milestones and the 1st is already done with 80% of time remaining. Demo's in Slack!"

L3

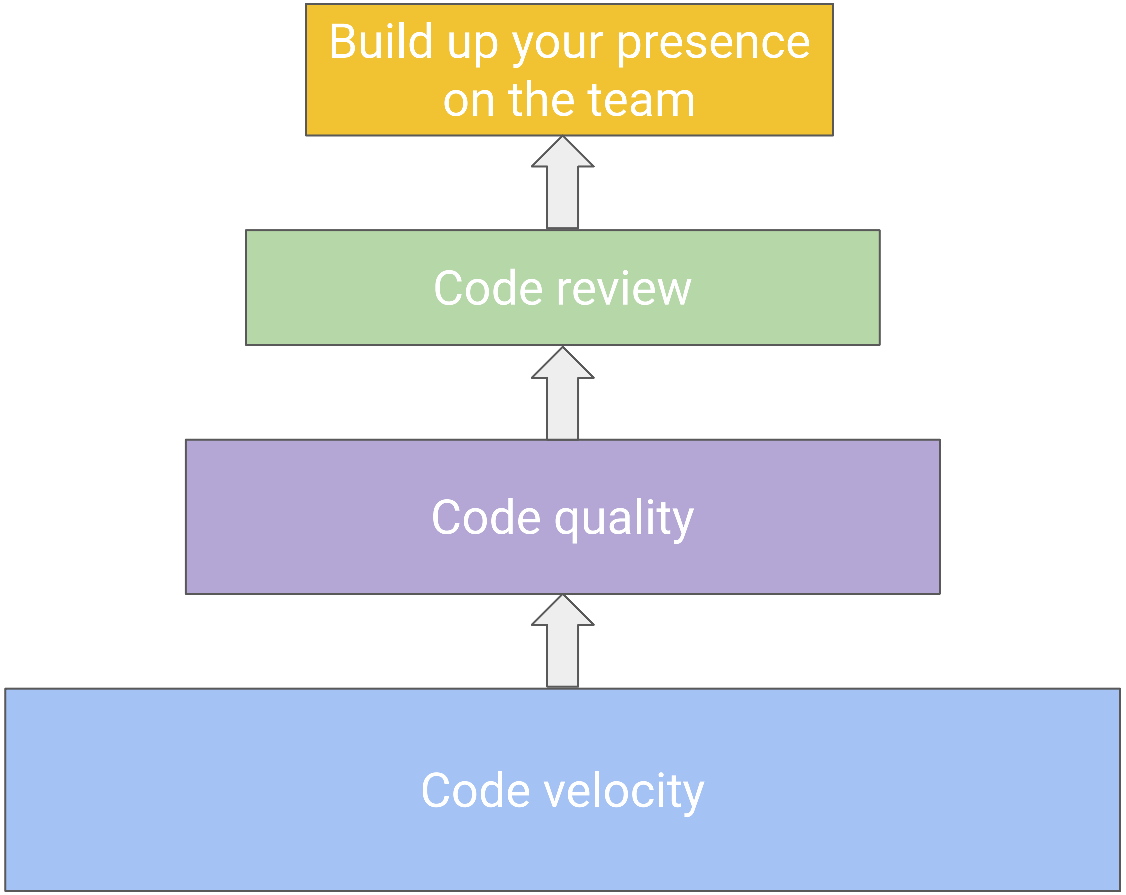
SWE II

L4

SWE III

Conclusion

High-Level Roadmap



This is NOT a strict ordering.

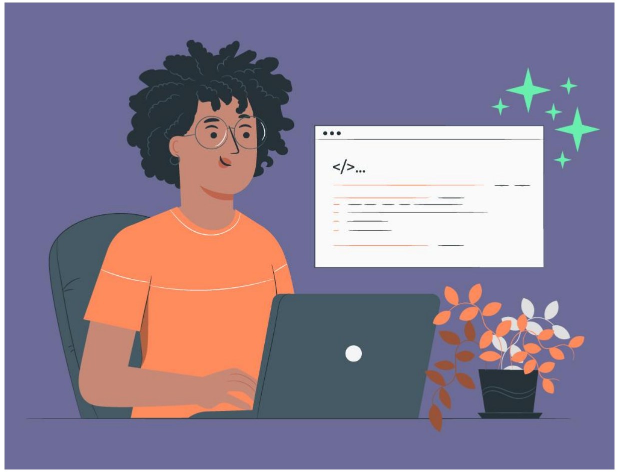
These phases will naturally blend into each other if you’re doing things properly.

Code Velocity 🏃

- **#1 priority for any engineer is to get stuff done at time**
- Move as fast as possible
- Only tackle low-hanging fruit for code quality:
  - Decompose tasks
  - Clean pull requests
- Don’t spend 2+ hours trying to find optimal approach

Code Quality 🌟

- Time to get to this phase: **3-6 months**
- By now, you will have gotten tons of feedback (mainly in code review)
- Now it makes sense spending 1-2 hours planning out your approach
    - Share it proactively



Code Review 🔍

- Time to get to this phase: **4-12 months**
- By now, you should be decent at writing good code
    - ...so help others do the same!
- Target pull requests from more junior engineers
- Learn from other rejections

Build Up Your Presence On The Team 🤝

- Time to get to this phase: **9-18 months**
- Several components:
  - Speaking up in meetings
  - Answering questions
  - Strengthening 1:1 relationships
- Switch from absorbing value to providing value

Done Is Better Than Perfect

DONE IS BETTER THAN PERFECT

BECAUSE PERFECT NEVER GETS DONE

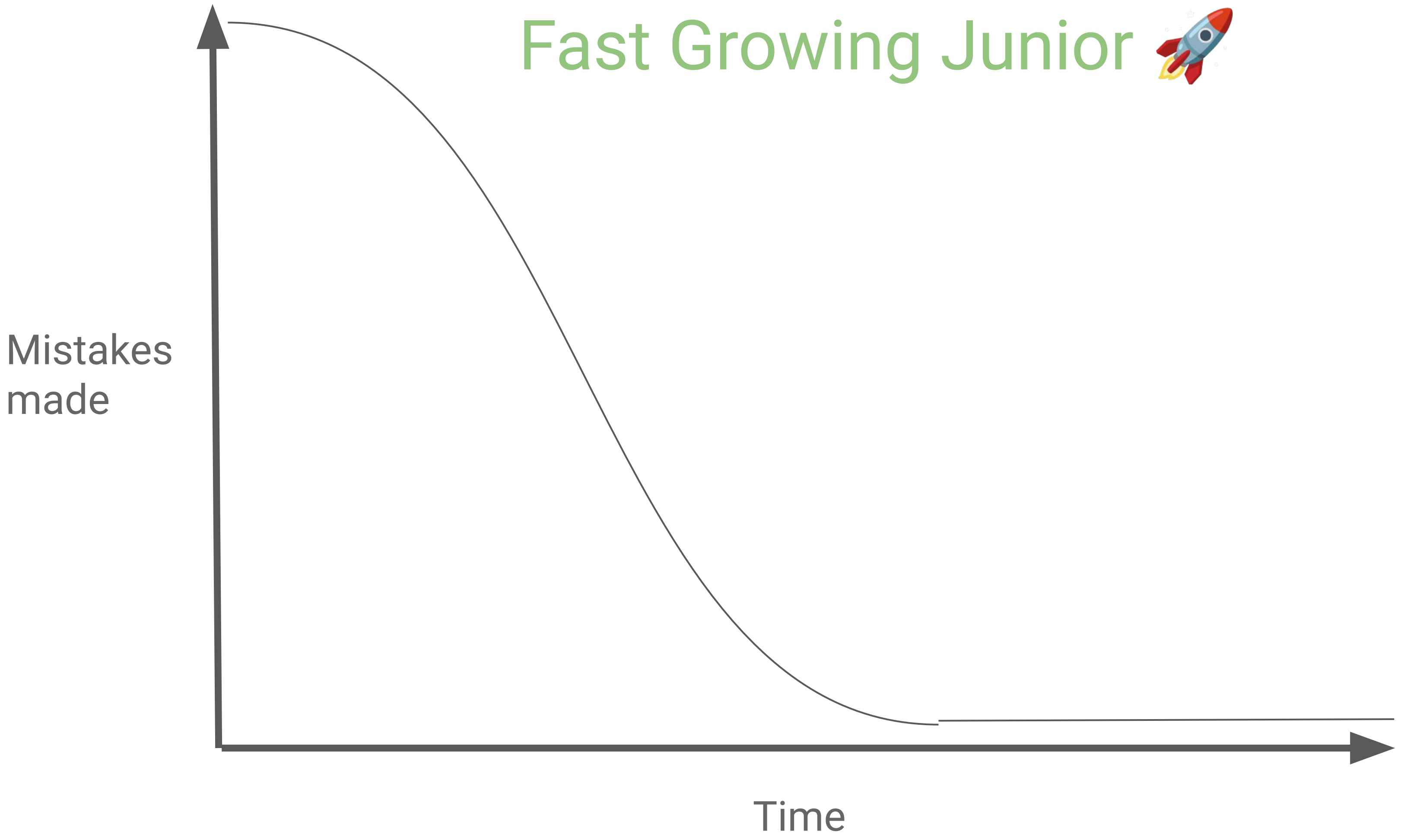
You Will Do *Many* Dumb Things As A Junior Engineer 🤦

- 🍝 Write messy code
- ⛓️‍💥 Break production
- 😕 Ask low-quality, confusing questions
- 👎 Say something completely wrong in a meeting
- ❌ Make a bad suggestion in code review

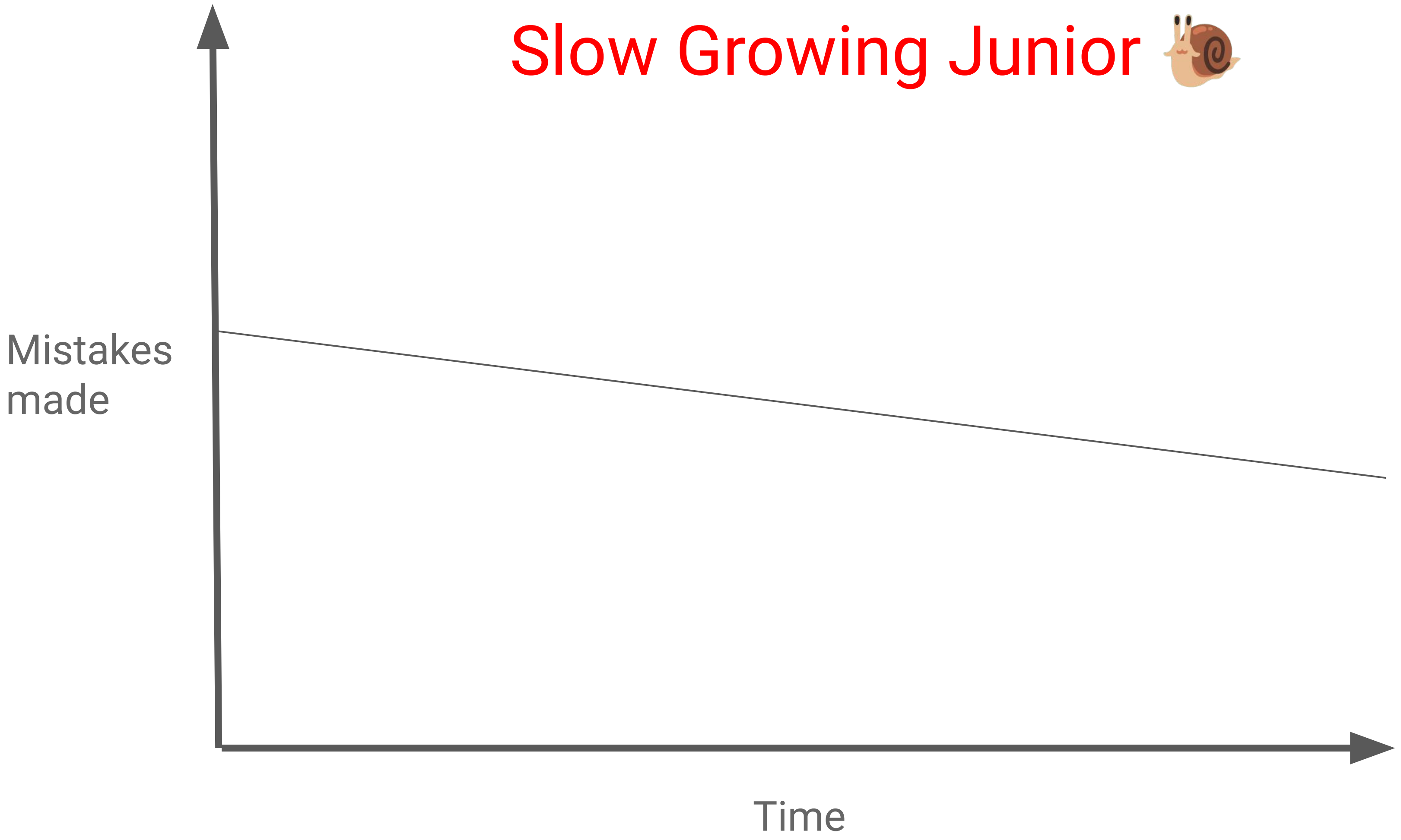
Do Dumb Thing -> Learn Why It's Dumb -> Repeat

- **Simple way to view L3 -> L4:** Get all the "dumb" things out of your system ASAP
    - Speed run feedbacks
- If you earn trust and communicate well, your team will be cool with this
- Lose the ego - *Embrace being a n00b*

Fast Growing Junior 🚀



Slow Growing Junior 🐌



Do dumb things fast. Embrace your mistakes.

Go Deeper: Follow Through

Become an expert craftsperson.

Submit clean pull requests.

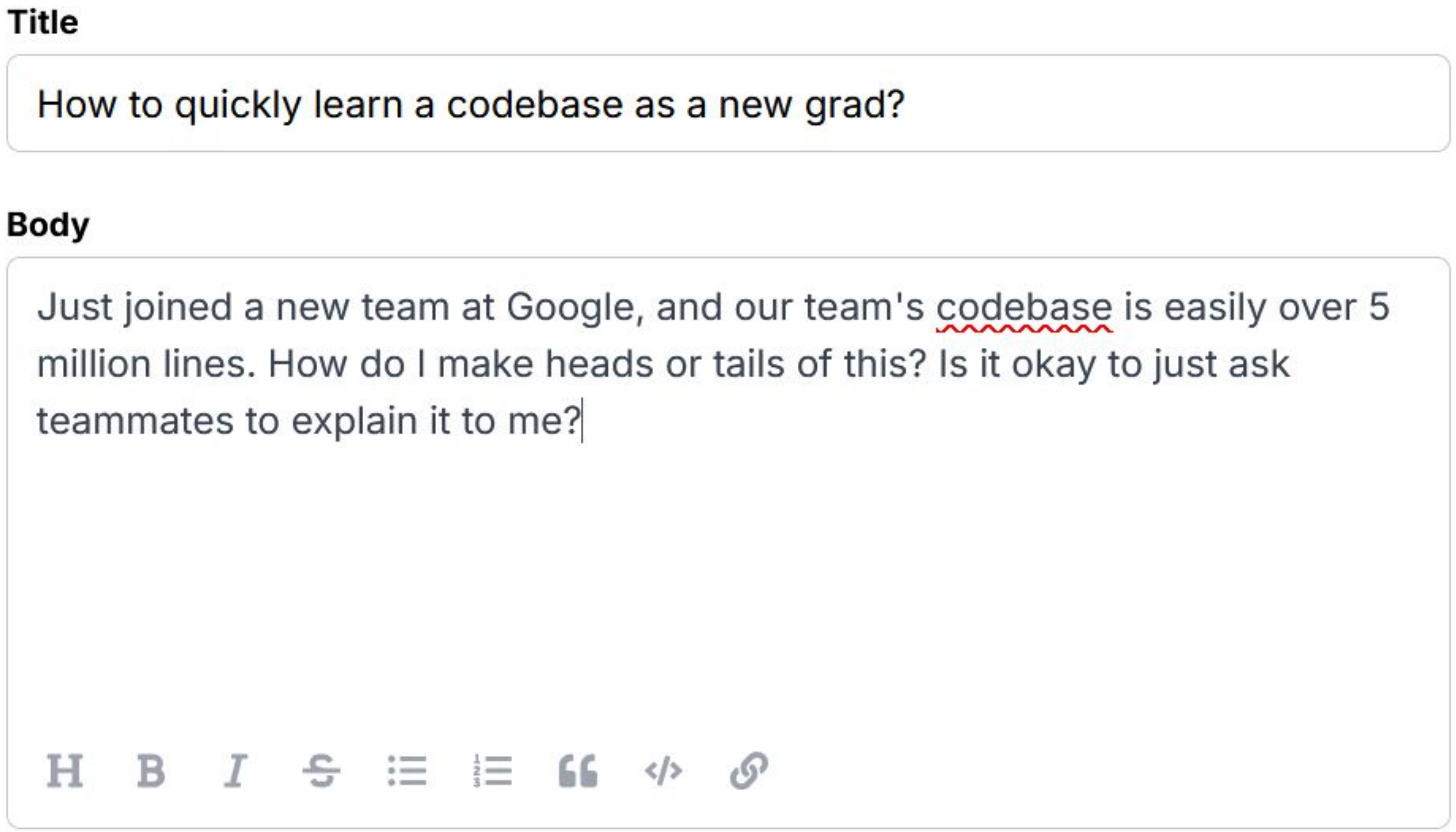
Do dumb things fast.

Think like an owner.

Feedback is a gift.

👨‍🏫 Professor Alex’s Homework For You

- ❑ Work with your manager to create a growth plan
- ❑ Apply code quality tactics to your pull requests
- ❑ Review 1 pull request for every 1 you submit
- ❑ Say at least 1 thing in every team meeting
- ❑ Set the agenda for your manager 1 on 1s
- ❑ Decompose your next task



Title

How to quickly learn a codebase as a new grad?

Body

Just joined a new team at Google, and our team's codebase is easily over 5 million lines. How do I make heads or tails of this? Is it okay to just ask teammates to explain it to me?

Level Up Your Code Quality As A Software Engineer

Ask Great Questions That Get Great Answers Quickly

These will get you 80%+ of the way to L4 📈

Nail Your Promotion As A Software Engineer

Become The Go-To Expert As A Software Engineer

Group Office Hours With Alex - Get Personalized Career Advice Privately

Event details

Monday, April 29, 2024 9:30am PDT to
Monday, April 29, 2024 10:30am PDT

Add to Google Calendar

Add to Apple Calendar

Add to Outlook Calendar

Taro Premium

This event has ended

22 people attended

thank you