

Good work isn't enough

**A clearer account of your contribution.
People who can vouch for it. Options
beyond your current role.**

Five questions for software engineers navigating
AI and change at work.

With examples and a worksheet for your next one-to-one.

You can do everything right and still lose your job

In 26 years at Microsoft, I've seen people do good work and still get laid off. Current skills offer no guarantee either.

When AI makes code quicker to produce, these questions matter more: who needs it, what improves, and who understands your contribution?

Choose useful work. Build evidence and relationships that help in your next team or job.

If you lose your job, that alone says nothing about the quality of your work.

Who actually uses it?

WEAK ANSWER

“We shipped it last quarter, and nobody’s complained.”

STRONGER ANSWER

“Eight of our twelve support teams use it each week to find account history during customer calls.”

Shipping tells you something exists. Find out who uses it, whether they return and what it helps them do.

All examples and numbers in this guide are hypothetical.

When you don't know who uses it

WHERE TO LOOK

Check active users, repeat use and task completion. For an API or internal service, identify the teams and systems that depend on it. Speak to a user.

IF YOU HAVE NO DATA

Ask your product partner or tech lead what you can track. Watch someone use it. For infrastructure or security work, ask what depends on it and what it protects.

IF HARDLY ANYONE USES IT

Ask why. Do people know it exists, can they access it, and what do they use instead? Agree whether to improve it, support adoption or stop investing. Check dependencies before retiring it.

Is this still a priority?

WEAK ANSWER

“I’m shipping everything assigned to me, and my code reviews are good.”

STRONGER ANSWER

“Customers abandon sign-up when it times out. My work tackles those failures, which supports our activation goal.”

Reliable delivery matters. You also need to understand why the team chose this work. Reliability, security and maintenance deserve a clear place in that conversation.

When the priority is unclear

WHERE TO LOOK

Check the current roadmap, incident reviews and customer support themes. Talk to your manager, tech lead or product partner about what has changed.

WHAT TO ASK

“What user problem or engineering risk does this address? What would happen if we delayed it?”

IF YOU STILL DON'T KNOW

Bring one piece of evidence and a choice: continue, reduce the scope or change direction. Agree the trade-off with the team. You should not have to set company strategy on your own.

What changed, and by how much?

WEAK ANSWER

“I closed 40 tickets and merged 25 pull requests.”

STRONGER ANSWER

“After our fix, sign-up timeouts fell from 8% to 2% of attempts over two weeks, at similar traffic levels.”

That connects the code to a result. Check what else changed before claiming credit, and explain your part in the team’s work.

When you don't have a number

WHERE TO LOOK

Check service dashboards, incident logs, cloud bills or build history. Ask the person who owns the data to help you interpret it.

A USEFUL MEASURE

Match it to the problem: failed sign-ups, recovery time, cost per request or build wait time. Record the starting point, period and volume. Check for regressions too.

IF THE DATA IS MISSING

Agree one measure and add tracking with the team. Until then, describe a verified incident or user example. For preventive work, explain the risk and test evidence. Don't invent savings or incidents avoided.

Who could recommend me?

WEAK ANSWER

“My manager can see how many tickets I close.”

STRONGER ANSWER

“The on-call lead saw me resolve an incident. Another team’s lead can explain how I helped them adopt our API.”

They can describe your judgment and how you work with others. Those relationships can help you find your next opportunity beyond this team.

When few people know your work

WHERE TO LOOK

Think of people who reviewed your design, handled an incident with you or use the service you maintain. Who understands your contribution beyond the ticket count?

SOMETHING TO SHARE

Write a short update: the problem, your contribution, the measured result and who helped. Share it where the relevant teams already work.

IF YOUR LIST IS SHORT

Ask your lead for a chance to pair on a related problem or present a useful finding. Follow up with a former teammate. Include reviews, mentoring and maintenance when you explain your work.

Is my learning helping?

WEAK ANSWER

“I use AI every day and generate twice as much code.”

STRONGER ANSWER

“Across ten similar changes, median time from starting work to merge fell from two days to one. Review effort and rework stayed level.”

Compare similar work and check what else changed. Include review, testing and maintenance when you judge whether faster code generation helps the team.

When you're unsure what to learn

WHERE TO LOOK

Ask where your team loses time or needs expertise. That might be debugging, system design, security or understanding the product, as well as using AI.

A SMALL TEST

Choose a bounded task with your lead. Try the skill on comparable work. Include review and testing time, then follow defects after release. Keep the same quality standard.

IF IT DOESN'T HELP

Check whether the task, tool or approach was a poor fit. Ask a peer to review the attempt. Choose your next learning step from what you found. Course completion alone won't answer this question.

My work and my next step

QUESTION	WHAT I KNOW	WHAT I'M ASSUMING
Who uses it or depends on it?		
Which priority or risk does it address?		
What changed, and by how much?		
Who could recommend me?		
How is my learning helping?		

My next action: _____

Who can help: _____ Review date: _____

Pick one gap to work on with your manager. Add the source and date beside your evidence. Keep confidential code, data and company records within your organisation.

You shouldn't have to guess alone

FOR MANAGERS

Engineering managers and leads need to explain shifting priorities and help people connect technical work to its value.

Make sure reviews, mentoring and the work that keeps systems running get credit too. Help engineers access evidence and find opportunities to grow.

Send this to your manager before your next one-to-one.

Try: "Could we work through this together? I'd like your help checking my assumptions and choosing a next step."