SPIN Interview Prep Worksheet

Your step-by-step system for turning conversations into job offers.

SPIN Selling was developed by Neil Rackham after studying 35,000+ sales calls. The best sellers don't pitch. They ask smart questions that help buyers realize they need to act. Replace 'buyer' with 'hiring manager' and 'solution' with 'you,' and you have a career pivot system.

Quick Reference: The SPIN Framework

Phase	Your Goal	Example Question	Common Trap
S - Situation	Contextualize	"I saw on the engineering blog you're migrating to AWS. How far along is the team?"	*Asking something you could have Googled in 5 minutes. Do your research first.*
P - Problem	Surface Pain	"With the migration happening, what's the biggest bottleneck for the junior devs?"	*Asking yes/no questions. Use "what" and "how" to get real answers.*
I - Implication	Show the Cost	"If that bottleneck persists, how does it impact your Q4 release cycle?"	*Nodding and moving on. Push deeper into the business cost.*
N - Need-Payoff	Bridge the Gap	"If a new hire handled the documentation and QA, would that speed up the migration?"	*Pitching yourself directly. Let them connect the dots.*

Step 1: Pre-Conversation Research

Complete this section BEFORE your informational interview, networking call, or job interview.

Target Company

Role / Team

Their Tech Stack (from job postings, engineering blog, LinkedIn)

Recent News / Changes (funding, product launches, reorgs, migrations)

3 Things I Know About Their Current Challenges

Step 2: Build Your SPIN Questions

Write your custom questions for this specific company and role. The best questions prove you did your homework and get the other person talking about real problems.

SITUATION Questions

Goal: Show you've done your homework. Confirm context, don't ask what you could Google.

Starters: "I noticed [specific thing from research]. Can you tell me more about..." | "I saw that [team/company] recently [specific change]. How is that going?" | "Your engineering blog mentioned [topic]. Where does that stand now?"

Q 1:

Q 2:

Q 3:

PROBLEM Questions

Goal: Surface real pain. Use open-ended 'what' and 'how' questions. Never yes/no.

Starters: "What's the most frustrating part of [process] right now?" | "Where do you see the biggest skill gaps on the team?" | "What keeps breaking that you wish someone would just fix?"

Q 1:

Q 2:

Q 3:

IMPLICATION Questions

Goal: Connect problems to business cost: time, money, morale, deadlines, missed goals.

Starters: "If [problem] persists, how does that impact [timeline/goal]?" | "When [issue] happens, what does that mean for the team's velocity?" | "If that gap stays unfilled for another quarter, what happens to [roadmap/OKRs]?"

Q 1:

Q 2:

Q 3:

NEED-PAYOFF Questions

Goal: Let THEM picture you in the role. Describe capabilities, not job titles.

Starters: "If someone could [capability from your background], would that help with [problem]?" | "Would it help to have someone who has experience [your transferable skill]?" | "If you had someone who could bridge [gap you identified], what would that unlock?"

Q 1:	
Q 2:	
Q 3:	

Step 3: Post-Conversation Debrief

Fill this out within 30 minutes of every conversation. Your notes decay fast.

Who did I talk to? (Name, title, team)

What problems did they mention?

What business costs did they connect to those problems?

Did they react positively to any of my Need-Payoff questions? Which ones?

What skills/capabilities seemed most valuable to them?

Follow-up action items (thank-you note, resource to share, next meeting)

30-Second SPIN Interview Checklist

Glance at this before you walk into any conversation.

Did I research their tech stack, blog, or recent news for my Situation questions?
Do I have 3 Problem questions that can't be answered with "Yes" or "No"?
Have I identified the business cost (Implication) for their most common pains?
Can I describe my value as a capability, not a job title?
Do I have a follow-up plan (thank-you note, resource to share, next conversation)?

Bonus: SPIN Templates for 5 Common Tech Roles

Use these as starting points. Customize every question to the specific company you're targeting.

DevOps / SRE	
S	"I see you're running Kubernetes on AWS. How large is the cluster?"
P	"What's the biggest pain point in your deployment pipeline right now?"
I	"If deployments keep taking that long, how does it affect your release cadence?"
N	"If someone streamlined CI/CD and reduced deployment time by 50%, what would that free up?"

QA / Test Engineering	
S	"What does your current test coverage look like across services?"
P	"Where are bugs slipping through to production most often?"
I	"When production bugs hit, what does that cost in engineering hours and customer trust?"
N	"If someone built out automated regression testing, how would that change your release confidence?"

Technical Writing	
S	"How does the team currently handle internal and external documentation?"
P	"What's the biggest gap in your docs right now?"
I	"When new engineers can't find answers in the docs, how long does onboarding take?"
N	"If someone owned the documentation system end-to-end, how would that impact team velocity?"

Project / Program Management	
S	"How many concurrent projects is the engineering team running right now?"
P	"Where do cross-team dependencies create the most friction?"
I	"When projects slip, what's the downstream impact on product launches?"
N	"If someone coordinated across teams and kept the roadmap on track, what would that unlock?"

Customer Success / Solutions Engineering	
S	"What does your customer onboarding process look like today?"
P	"Where are customers getting stuck or churning?"
I	"If churn stays at this rate, what does that mean for your ARR targets?"
N	"If someone bridged the gap between product and customers to reduce time-to-value, how would that change retention?"