interview101

PERSONALIZED INTERVIEW PLAYBOOK

Your Roadmap to Landing This Role

Everything you need to walk into your interview confident, prepared, and ready to win.

PREPARED FOR
Marcus Webb

TARGET ROLE
SWE

COMPANY
Meta

GENERATED
May 04, 2026

1

Your Interview Prep Starts Here

A quick snapshot of where you stand and how to use this report.

YOUR QUICK ASSESSMENT

You bring a strong technical foundation that aligns well with Meta's engineering standards, with your deep technical background serving as your most compelling asset for succeeding as a Software Engineer at the company.

To succeed as a Meta SWE, prioritize deepening your system design fundamentals—this is where you'll create the most impact. Pairing targeted practice here with your solid coding skills will position you competitively for the next round.

What's Inside

Short on Time?
30-minute prep path

- → Read Where You Stand (Section 2)
- → Memorize your pitches (Section 4)
- → Review your top 3 stories (Section 5)
- → Grab the cheat sheet (Section 10)

Full Preparation
2-hour deep dive

- → Read the full report front-to-back
- → Practice all stories out loud
- → Role-play the top 5 questions
- → Customize your questions to ask

2

Where You Stand

An objective assessment of your fit for this role, based on your resume and the job requirements.

YOUR FIT ASSESSMENT

Competitive

Your proven expertise scaling messaging infrastructure positions you well for this role, though you'll want to build C++ proficiency to fully own infrastructure systems at Meta's scale.

75 / 100

Skills match — 28 / 40

Strong match on Python, Go, Java, distributed systems, API design, and message queuing; missing C++ and system design depth.

Experience alignment — 35 / 35

Five years with senior-level payments infrastructure at Stripe perfectly aligns with WhatsApp's distributed messaging system requirements.

Culture & role fit — 12 / 25

Resume doesn't yet signal Meta's collaboration focus or emerging technology adoption; current framing emphasizes individual technical ownership.

Your Strengths

Proven Large-Scale Messaging Infrastructure Expertise

Your experience building high-volume messaging systems at Stripe and Twilio directly aligns with WhatsApp's infrastructure needs. At Twilio, you built SMS delivery pipelines handling 500M+ messages monthly, while at Stripe you owned distributed systems processing 1B+ API calls. **This background in massive-scale message processing** positions you perfectly for "design and build distributed backend systems for WhatsApp messaging infrastructure" where reliability and scale are paramount.

End-to-End Ownership with Production Excellence

Your track record demonstrates exactly the "ownership and accountability" Meta values, with concrete evidence of production excellence. You owned Stripe's distributed rate-limiting service handling 2M+ requests/second with sub-3ms latency, led zero-downtime migrations, and reduced MTTR from 45 to 12 minutes through improved operational practices. **This proven ability to own features end-to-end** matches the JD requirement to "own features from design through production deployment and monitoring."

Technical Leadership Through Mentorship Impact

Your mentorship experience at Stripe, where you guided 3 junior engineers and conducted 50+ technical interviews, demonstrates the "mentorship and engineering excellence" Meta seeks. Combined with your open-source contributions to Twilio's Python library (2K+ GitHub stars), you've shown commitment to developing others and advancing engineering practices. **This leadership foundation** aligns perfectly with the responsibility to "mentor junior engineers and contribute to engineering excellence."

Gaps to Address

No C++ Experience for Infrastructure Role

The job description lists C++ as a required skill alongside Python, Go, and Java for WhatsApp's messaging infrastructure. Your resume shows strong Python and Go expertise from Stripe, but **completely lacks any C++ experience or projects**. Given WhatsApp's scale (billions of messages daily), performance-critical components likely use C++ for optimization. While your distributed systems experience is solid, you'll need to address this language gap and demonstrate how your systems thinking translates across languages.

→ See Section 5 for stories to bridge this

Limited Cross-Functional Collaboration Evidence

The role emphasizes "collaborate with cross-functional teams to define technical requirements and architecture" as a core responsibility. Your resume focuses heavily on technical implementation and infrastructure ownership, but **provides minimal evidence of working with product, design, or business stakeholders**. Your bullets mention mentoring engineers and conducting interviews, but don't showcase the cross-team partnership skills Meta values for defining requirements and architecture decisions with non-technical partners.

→ See Section 5 for stories to bridge this

No AI Coding Tools Experience

The job description specifically mentions "engage with AI-assisted coding tools Meta is rolling out across engineering" and lists "experience with AI coding tools" as a preferred skill. Your technical background is strong, but **your resume shows no exposure to AI-powered development tools** like GitHub Copilot, CodeWhisperer, or similar platforms. Given Meta's emphasis on emerging technology adoption and their investment in AI tooling, this gap could signal you're behind on modern development practices. → See Section 5 for stories to bridge this

YOUR PREP PRIORITY

Here's how to use this report

You're walking in with world-class infrastructure expertise and proven ownership that Meta desperately needs—your messaging systems work at scale. Your critical gap is demonstrating culture fit and cross-functional collaboration, which Section 5 addresses through STAR stories Meta will ask about directly. Prioritize building three stories around working across Product, Design, and Data teams to show you're not just a technical isolate. Second, spend 30 minutes in Section 3 decoding what Meta specifically means by "collaborative engineer" in this JD so you can mirror that language in every answer. When you open this report, go straight to Section 4 to lock in your core pitch, then immediately move to Section 5 to script collaboration moments—that's your confidence builder and your competitive edge.

3

What They Actually Want

The job description tells part of the story. Here's what's really driving this hire.

Reading Between the Lines

WHAT THEY SAID	WHAT THEY MEAN	HOW YOU DEMONSTRATE IT
"Design and build distributed backend systems for WhatsApp messaging infrastructure"	You'll architect systems handling billions of messages daily with extreme scale, reliability, and performance requirements under Meta's infrastructure standards.	Your **distributed rate-limiting service** handling 2M+ requests/second shows relevant scale experience, plus your idempotency layer across 1B+ API calls demonstrates messaging reliability patterns.
"Own features end-to-end from design through production deployment and monitoring"	Meta evaluates ownership depth - can you drive complex projects independently while maintaining production excellence throughout the entire lifecycle.	Your **payment retry system ownership** shows end-to-end delivery, plus your on-call rotation leadership demonstrates production monitoring and incident response capabilities.
"Participate in code review, on-call rotations, and production incident response"	Meta expects E4+ engineers to handle production pressure, reduce MTTR efficiently, and maintain system reliability through disciplined operational practices.	Your **MTTR reduction from 45 to 12 minutes** directly shows incident response excellence, plus your current on-call rotation ownership demonstrates operational maturity.
"Mentor junior engineers and contribute to engineering excellence"	Meta assesses behavioral impact - can you elevate team performance, scale knowledge effectively, and drive engineering standards beyond individual contributions.	Your **mentoring of 3 junior engineers** plus conducting 50+ technical interviews shows direct people development experience and contribution to team hiring standards.
"Engage with AI-assisted coding tools Meta is rolling out"	Meta is prioritizing AI adoption across engineering - they want engineers who can adapt to new tooling and leverage AI for productivity gains.	You lack direct AI coding tool experience, but your **open-source contribution** and technical interview leadership show adaptability to new technologies and tooling adoption.

Why This Role Exists

WhatsApp processes over 100 billion messages daily across 2+ billion users, making it one of the world's largest distributed systems. Meta's massive scale demands engineers who can build infrastructure that doesn't just work, but scales seamlessly under unprecedented load. The emphasis on distributed systems and end-to-end ownership reflects the technical complexity of maintaining real-time messaging infrastructure at this magnitude.

The combination of multiple programming languages (Python, Go, C++, Java) and emphasis on mentoring suggests a team managing legacy systems while modernizing architecture. The specific callout of AI coding tools indicates Meta is actively transforming how engineering work gets done. The focus on production excellence and incident response signals this team has experienced scaling pains and needs engineers who can maintain stability during rapid growth.

The pain they're solving: WhatsApp's infrastructure is hitting complexity limits that require **architectural modernization without breaking the world's largest messaging platform**. They need engineers who can refactor critical systems in production, mentor teams through technical transitions, and leverage AI tools to accelerate development velocity. Position yourself as someone who thrives on high-stakes technical challenges and can drive system evolution while maintaining bulletproof reliability.

Company Intelligence That Matters

META CORE VALUES IN PLAY

Every interview round evaluates alignment to these values. For this role, focus on:

- Move Fast — shipped a working solution quickly under ambiguity, removed blockers proactively
- Be Bold — proposed or drove a technically risky decision that paid off
- Focus on Long-Term Impact — made a technical decision that prioritised system health or scalability over short-term speed
- Be Open — drove cross-team alignment through transparency, shared context, or public technical documentation
- Build Social Value — engineering decision rooted in genuine user benefit at scale

Have a STAR story ready for each. Vague answers are the #1 reason qualified candidates fail.

INTERVIEW PROCESS

Process includes recruiter screen, 45-minute technical screen with two coding problems, then onsite with two additional coding interviews, behavioral interview covering Product Sense/Analytical Thinking/Leadership & Drive, and systems design interview. All technical rounds assess algorithms, data structures, and system design thinking.

CULTURE SIGNAL

Meta operates with radical transparency and intellectual honesty—engineers are expected to challenge decisions openly, regardless of hierarchy. The flat structure means you'll constantly collaborate across teams without formal authority, requiring influence through technical merit and clear communication rather than position.

WHAT SUCCESS LOOKS LIKE

Top performers own features end-to-end from architecture to production monitoring, actively participate in code reviews and on-call rotations, mentor others through technical guidance, and drive cross-functional alignment through influence. They move fast on high-impact initiatives while maintaining production excellence.

What Makes This Interview Different

2026 AI-ASSISTED CODING ROUND

As of October 2025, Meta has introduced an AI-assisted coding round that replaces one of the two traditional coding rounds at the onsite. You work in a specialised CoderPad environment with an AI assistant built in. The AI can help with syntax and boilerplate — but it cannot solve the problem for you. Code execution is OFF. Interviewers evaluate whether you can direct, validate, and own the AI output — not whether AI can code for you. Using the AI is optional; what matters is that the solution is yours. This round is now standard for all SWE roles in 2026. Additionally, Meta uses a hiring committee model — a cross-functional group reviews all feedback after the loop. No single interviewer has veto power, meaning consistency across all rounds matters as much as any single strong performance.

Hidden Priorities (What the JD Reveals)

1. **End-to-End Production Ownership Beyond Code Writing** JD signal

   The JD emphasizes 'Own features end-to-end from design through production deployment and monitoring' and includes 'on-call rotations and production incident response.' This signals they want engineers who **take full accountability for operational excellence**, not just feature delivery.

2. **Cross-Team Influence Without Direct Authority** JD signal

   The JD leads with 'Collaborate with cross-functional teams' and emphasizes 'Mentor junior engineers' alongside technical delivery. This suggests you'll need to **drive results through influence** across multiple teams rather than working in isolation.

3. **Ship Fast Then Iterate Philosophy Assessment** Company intel

   Meta's 'Move fast' principle and engineering culture of 'moving fast, breaking things' means interviews will evaluate whether you can **deliver working solutions quickly** rather than over-engineering initial implementations. They prioritize speed to production over perfect first attempts.

⚠ Watch Out For These Mistakes

Downplaying Cross-Team Collaboration Impact

Meta's flat structure requires constant cross-team work, but you might focus too heavily on individual technical achievements. When discussing your payment retry system or API gateway work, emphasize how you **influenced teams you didn't manage** and drove consensus across engineering groups. Use Section 5 stories to highlight collaborative problem-solving rather than solo technical execution.

Avoiding System Design Trade-offs

Your distributed systems experience is strong, but Meta interviewers expect you to discuss architectural trade-offs openly. Don't just explain your rate-limiting service's performance—discuss **what you sacrificed to achieve it** and alternative approaches you considered. Reference Section 7 gap scripts to prepare for deeper system design discussions about scalability versus consistency choices.

Underselling Product-Minded Engineering Decisions

Meta values engineers who think like product owners, but you might frame your work purely in technical terms. When discussing your 94% reduction in duplicate charges, connect it to **user experience and business impact**. Explain how technical decisions like your idempotency layer directly improved developer experience and merchant trust, not just system reliability.

WHAT THIS MEANS FOR YOUR INTERVIEW

- → **Demonstrate cross-team influence** — Share examples where you drove technical decisions across teams without formal authority, showing how you built consensus through technical merit.
- → **Emphasize end-to-end ownership** — Describe projects where you owned the full lifecycle from design to production monitoring, including how you handled incidents and performance optimization.
- → **Show mentorship impact** — Provide specific examples of how you've elevated junior engineers through code reviews, technical guidance, or knowledge sharing initiatives.
- → **Balance speed with scale** — Highlight instances where you moved fast on high-priority features while building systems that could handle long-term growth and reliability requirements.
- → **Practice distributed systems design** — Prepare to discuss WhatsApp-scale challenges like message routing, data consistency, and handling billions of daily messages across global infrastructure.

4

Your Story, Interview-Ready

Polished answers to "Tell me about yourself" — the most predictable question, and the easiest to fumble.

The 30-Second Pitch

~30 seconds

> I'm a Senior Software Engineer with 5 years building distributed systems at Stripe and Twilio. I designed Stripe's payment retry idempotency layer that **reduced duplicate charges by 94% across 1B+ monthly API calls** and own their rate-limiting service handling **2M+ requests/second with p99 latency under 3ms**. This experience building resilient messaging infrastructure directly translates to **WhatsApp's scale challenges**. I'm excited to move fast on systems that connect billions globally.

1. **Hook:** Establishes credibility immediately with specific experience
2. **Proof:** Concrete numbers make it real and memorable
3. **Bridge:** Connects your past to their specific opportunity
4. **Close:** Shows ambition without sounding desperate

The 2-Minute Version

~2 minutes

> **Past:** I started at Twilio building SMS delivery infrastructure that **handled 500M+ messages monthly**. Early on, I owned carrier failover logic that **improved global delivery rates from 94.2% to 97.8%** — real impact across distributed systems.
>
> **Present:** At Stripe, I've **owned payments infrastructure serving 1B+ monthly API calls**. My biggest win was designing an idempotency layer that **reduced duplicate charge incidents by 94%** and building a rate-limiting service **handling 2M+ requests/second** with sub-3ms latency.
>
> **Pivot:** I want to tackle messaging infrastructure at **global WhatsApp scale** — the distributed systems challenges that come with billions of users.
>
> **Future:** Meta's focus on **end-to-end ownership and cross-team impact** matches exactly how I work. I'm excited to ship messaging infrastructure improvements and mentor engineers while **adopting Meta's AI-assisted coding tools**.

"Why This Company?"

I'm drawn to Meta's **Move Fast** culture and engineering-first approach. Having owned distributed systems at Stripe handling billions of requests, I'm excited to tackle the unique constraints of **serving 3 billion people**. Meta's flat structure aligns with my experience leading through influence—mentoring engineers and driving infrastructure improvements without formal authority. The opportunity to build awesome things at unprecedented scale energizes me.

Tip: This connects the Move Fast LP and scale anchor to his distributed systems ownership and mentoring experience, showing cultural alignment.

"Why this role specifically?"

What draws me to this role is the opportunity to **"design and build distributed backend systems"** at WhatsApp's scale. At Stripe, I built distributed systems handling 2M+ requests/second with p99 latency under 3ms, plus designed idempotency systems across 1B+ monthly API calls. The focus on **"production excellence"** and **"on-call rotations"** aligns perfectly with my experience owning payments infrastructure on-call, where I reduced MTTR from 45 to 12 minutes.

Tip: Mirror exact JD phrases like "design and build distributed backend systems" and "production excellence" while connecting each to specific resume metrics and company achievements.

Delivery Tips

- ✓ **Practice out loud** — reading silently isn't the same. Record yourself and listen back.
- ✓ **Pause after key points** — let your numbers land. Important stats deserve a beat.
- ✓ **End with forward energy** — your last sentence should make them want to hear more.
- ✗ **Don't memorize word-for-word** — know the beats, not the script. Robotic delivery kills credibility.
- ✗ **Don't rush the "why here"** — this is where you show genuine interest. Slow down.
- ✗ **Don't apologize for gaps** — not here. Save objection handling for when they ask directly.

5

6 Stories That Win Interviews

Your proof points — built from your resume, ready to personalize and deliver.

How These Stories Work

We've built STAR stories from your resume — but **only you know the full details**. Each story has three parts:

✓ **Verified** = Facts directly from your resume (company, role, metrics)
◂ **Draft** = Plausible details we've inferred — **review and correct these**
○ **You fill in** = Details only you know — add these before your interview

From your resume Draft — verify this You fill in

Before your interview: Read each story, correct anything we got wrong, and fill in the blanks. Practice telling each story in 2-3 minutes. The goal isn't to memorize — it's to know the beats so you can deliver naturally.

YOUR 6 STORIES

1. Payment Idempotency System
2. Zero-Downtime Migration
3. Incident Response Optimization
4. Engineering Team Growth
5. API Gateway Reliability
6. SMS Delivery Optimization

1 Payment Idempotency System

Build awesome things | Focus on long-term impact

✓ FROM RESUME What we know for certain

"Designed and built idempotency layer for Stripe's payment retry system — reduced duplicate charge incidents by 94% across 1B+ monthly API calls"

USE THIS STORY FOR

Tell me about a time you built something impactful / Describe a complex technical project you led / How do you approach system design

DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> At Stripe in *[what year?]*, our payment retry system was creating a massive problem for merchants. When network issues or temporary failures occurred, *[what was happening - multiple retry attempts? customer complaints?]*, and we were processing over 1 billion API calls monthly. *[What was the business impact you were seeing?]*

TASK VERIFY

> I needed to design and build a comprehensive idempotency layer that would prevent duplicate charges while maintaining the reliability merchants expected. *[Was this your idea or assigned to you? What was the timeline?]*

ACTION VERIFY

> I *[designed what kind of architecture - distributed cache? database solution?]* and implemented *[what were your key technical decisions?]*. *[How did you handle edge cases? What was your rollout strategy? Did you work with other teams?]* I also *[how did you test this at scale?]*.

RESULT FROM RESUME

> The idempotency layer reduced duplicate charge incidents by 94% across 1B+ monthly API calls. *[What happened next? Recognition from leadership? Became standard pattern? How did merchants react? Any promotions or expanded scope?]*

Before You Use This Story

- [ ] Add the 'before' metric — what was the duplicate charge rate before your fix?
- [ ] Specify the technical architecture you chose and why
- [ ] Include the rollout timeline and any challenges during deployment
- [ ] Clarify your role vs team contributions in this project

Likely Follow-up Questions — Prepare Your Answers

"How did you test this system at billion-call scale?"

> *Focus on your testing strategy, load testing approach, and how you validated correctness without impacting production*

"What trade-offs did you consider in your design?"

> *Discuss performance vs consistency, storage costs, complexity vs reliability - show your systems thinking*

"How did you handle edge cases like race conditions?"

> *Get technical about concurrent requests, network partitions, and failure scenarios you anticipated*

2 Zero-Downtime Migration

Move fast | Focus on long-term impact

✓ FROM RESUME What we know for certain

"Led migration of legacy Python 2 payment processing pipeline to Python 3 with zero downtime across 18 months"

USE THIS STORY FOR

Tell me about a time you had to balance speed with reliability / Describe a challenging migration project / How do you handle technical debt

‹ DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> At *[company]*, our critical payment processing pipeline was running on Python 2, which *[was reaching end of life? causing specific problems?]*. The system handled *[what volume/criticality?]* and any downtime would *[what would be the business impact?]*.

TASK VERIFY

> I was tasked with leading the migration to Python 3 over 18 months while maintaining zero downtime. *[Was this your proposal or mandated? What made this particularly challenging - system complexity? team size? dependencies?]*

ACTION VERIFY

> I developed *[what was your migration strategy - gradual rollout? dual-stack approach?]* and *[how did you coordinate with other teams? What was your testing approach?]*. *[How did you handle dependencies? What was your rollback plan?]* Throughout the 18 months, I *[how did you track progress and manage risks?]*.

RESULT FROM RESUME

> Successfully completed the migration with zero downtime across 18 months. *[What happened after completion? Performance improvements? Team recognition? Did this become a template for other migrations? What was the long-term impact?]*

Before You Use This Story

- [ ] Specify why this took 18 months - what made it so complex?
- [ ] Add details about your migration strategy and phasing approach
- [ ] Include metrics beyond uptime - performance, reliability improvements
- [ ] Clarify the team size and cross-functional coordination required

Likely Follow-up Questions — Prepare Your Answers

"Why did this take 18 months instead of doing it faster?"

> *Explain the technical and business constraints that drove the timeline - system complexity, risk management, resource allocation*

"How did you ensure zero downtime throughout?"

> *Detail your deployment strategy, monitoring, rollback procedures, and how you validated each phase*

"What was the biggest risk you encountered during migration?"

> *Share a specific challenge that threatened the timeline or stability, and how you mitigated it*

3 Incident Response Optimization

Move fast | Meta, metamates, me

✓ FROM RESUME What we know for certain

"On-call rotation owner for payments infrastructure — reduced MTTR from 45 minutes to 12 minutes through improved runbooks and alerting"

USE THIS STORY FOR

Tell me about a time you improved a process / Describe how you handle production incidents / How do you drive operational excellence

DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> As the on-call rotation owner for payments infrastructure at *[company]*, our team was struggling with *[what types of incidents were common?]*. Our mean time to resolution was 45 minutes, which meant *[what was the business impact - lost revenue? customer complaints?]*.

TASK VERIFY

> I needed to systematically improve our incident response process to reduce MTTR and *[were there other goals - reduce stress? improve team confidence?]*. *[Was this self-initiated or assigned? What resources did you have?]*

ACTION VERIFY

> I *[what specific changes did you make to runbooks - standardized format? added troubleshooting steps?]* and overhauled our alerting by *[what alerting improvements - better signal-to-noise? automated diagnostics?]*. *[How did you get team buy-in? Did you provide training? What tools did you implement?]*

RESULT FROM RESUME

> Reduced MTTR from 45 minutes to 12 minutes through improved runbooks and alerting.
> *[What was the team's reaction? Did this reduce on-call burden? Any recognition from leadership? Did other teams adopt your approach?]*

Before You Use This Story

- [ ] Specify what types of incidents were most common and time-consuming
- [ ] Detail the specific improvements you made to runbooks and alerting
- [ ] Add context about team size and how you measured the 45→12 minute improvement
- [ ] Include any tools or automation you implemented

Likely Follow-up Questions — Prepare Your Answers

"What was causing the long resolution times originally?"

> *Analyze the root causes - poor documentation, unclear escalation, inadequate monitoring - and how you diagnosed this*

"How did you get the team to adopt new processes?"

> *Discuss change management, getting buy-in, training, and making processes stick during high-stress incidents*

"How do you measure and maintain these improvements over time?"

> *Talk about metrics tracking, regular reviews, and preventing regression as team membership changes*

4 Engineering Team Growth

Meta, metamates, me | Be direct and respect your colleagues

✓ FROM RESUME What we know for certain

"Mentored 3 junior engineers; conducted 50+ technical interviews for the infrastructure team"

USE THIS STORY FOR

Tell me about a time you mentored someone / How do you help others grow / Describe your approach to giving feedback

◦ DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> At *[company]*, our infrastructure team was growing rapidly and *[what was the context - hiring surge? team expansion? knowledge gaps?]*. We had *[how many junior engineers joined? what was their background?]* and I was *[how were you selected for this mentoring role?]*.

TASK VERIFY

> I needed to mentor 3 junior engineers to help them *[what were their specific growth areas?]* while also conducting 50+ technical interviews to *[what was the hiring goal? what roles were you interviewing for?]*. *[What was the timeline for this?]*

ACTION VERIFY

> For mentoring, I *[what was your approach - regular 1:1s? code reviews? project assignments?]* and *[how did you tailor your approach to each person's needs?]*. For interviewing, I *[what was your interview process? how did you ensure consistency? did you help design the questions?]* *[How did you balance this with your regular engineering work?]*

RESULT FROM RESUME

> Successfully mentored 3 junior engineers and conducted 50+ technical interviews for the infrastructure team. *[What happened with your mentees - promotions? successful projects? positive feedback? What was the hiring success rate? Did any of your processes get adopted more widely?]*

Before You Use This Story

- [ ] Add specific outcomes for your mentees - promotions, successful projects, skill development
- [ ] Include details about your mentoring approach and how you tracked progress
- [ ] Specify the interview success rate and quality of hires
- [ ] Clarify the time period over which this mentoring and interviewing occurred

Likely Follow-up Questions — Prepare Your Answers

"How did you tailor your mentoring approach to different people?"

> *Share specific examples of different mentees' needs and how you adapted your style and methods*

"What's your philosophy on giving difficult feedback?"

> *Discuss how you balance being direct with being supportive, and share an example of challenging feedback you delivered*

"How do you evaluate technical skills in interviews?"

> *Explain your interview framework, what you look for beyond coding ability, and how you ensure fair evaluation*

5 API Gateway Reliability

Build awesome things | Focus on long-term impact

✓ FROM RESUME What we know for certain

"Built versioned API gateway in Go serving Stripe's external developer API — 99.999% uptime over 18 months"

USE THIS STORY FOR

Tell me about a time you built something reliable / How do you ensure high availability / Describe your approach to API design

◦ DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> At *[which company?]* in *[what year?]*, our external developer API was experiencing *[what types of reliability issues? outages? performance problems?]* that were frustrating our developer community. *[What was the existing architecture? Was there an old gateway system or direct service calls?]*

TASK VERIFY

> I was tasked with building a new versioned API gateway in Go that could *[what were the specific requirements beyond uptime? traffic volume? feature support?]*. The goal was to create a rock-solid foundation for our external developer ecosystem.

ACTION VERIFY

> I designed the gateway with *[what specific reliability patterns? circuit breakers? health checks? monitoring?]*. *[What was your approach to versioning? How did you handle backwards compatibility?]* I implemented *[what specific Go frameworks or libraries?]* and set up *[what monitoring and alerting systems?]*.

RESULT FROM RESUME

> The API gateway achieved 99.999% uptime over 18 months serving Stripe's external developer API. *[What happened after launch? Developer feedback? Team recognition? Did this become the template for other services? Any promotions or expanded responsibilities?]*

Before You Use This Story

- [ ] What was the 'before' uptime percentage that you improved from?
- [ ] Add specific technical details about your reliability architecture (circuit breakers, load balancing, etc.)
- [ ] Include the scale metrics - how many requests per second or developers served?
- [ ] Describe the versioning strategy and how you handled API evolution

Likely Follow-up Questions — Prepare Your Answers

"How did you achieve 99.999% uptime? What specific architectural decisions contributed to that reliability?"

> *Walk through your reliability engineering approach - monitoring, redundancy, graceful degradation, deployment strategies. Show systems thinking.*

"How did you handle API versioning and backwards compatibility for external developers?"

> *Demonstrate product sense and developer empathy. Explain your deprecation strategy and how you communicated changes.*

"What was the most challenging technical problem you solved while building this gateway?"

> *Pick a specific deep technical challenge that shows problem-solving skills. Walk through your debugging and solution process.*

6 SMS Delivery Optimization

Move fast | Focus on long-term impact

✓ FROM RESUME What we know for certain

"Designed carrier failover logic that improved global delivery rates from 94.2% to 97.8%"

USE THIS STORY FOR

Tell me about a time you solved a complex technical problem / How do you approach system optimization / Describe a time you improved user experience through engineering

◦ DRAFT Plausible story — review and personalize

We've written a realistic version based on your resume. Read through it, correct anything that's wrong, and fill in the blanks marked with [brackets].

SITUATION VERIFY

> At *[which company?]* in *[what timeframe?]*, our SMS delivery system was underperforming with only 94.2% global delivery rates. *[What was causing the failures? Regional carrier issues? Network problems?]* This was impacting *[what type of messages? user authentication? notifications?]* for our users worldwide.

TASK VERIFY

> I needed to design a carrier failover system that could *[what were the specific technical requirements? real-time switching? cost optimization? regional coverage?]*. The challenge was improving reliability while *[maintaining cost efficiency? meeting latency requirements?]*.

ACTION VERIFY

> I analyzed *[what data sources did you use? delivery logs? carrier performance metrics?]* to understand failure patterns. I designed a failover logic that *[how did it work? priority-based? geographic? cost-weighted?]*. *[What technologies did you use? How did you implement the switching logic? What was your testing approach?]*

RESULT FROM RESUME

> The carrier failover logic improved global delivery rates from 94.2% to 97.8%. *[What was the impact on users? Reduced support tickets? Improved conversion rates? Did this system get adopted by other teams? Any recognition or career impact?]*

Before You Use This Story

- [ ] Add the volume scale - how many SMS messages per day/month were you handling?
- [ ] Include specific technical details about your failover algorithm and decision criteria
- [ ] Quantify the business impact beyond delivery rates (cost savings, user satisfaction, etc.)
- [ ] Describe how you measured and validated the improvement across different regions

Likely Follow-up Questions — Prepare Your Answers

"How did you analyze the SMS delivery failures and decide on your failover strategy?"

> *Show your analytical process - data gathering, pattern identification, hypothesis testing. Demonstrate systematic problem-solving.*

"How did you balance delivery rates with cost when designing the failover logic?"

> *Explain the tradeoffs you considered and how you optimized for multiple objectives. Show business understanding alongside technical skills.*

"How did you test and validate this system before rolling it out globally?"

> *Describe your testing methodology, gradual rollout strategy, and monitoring approach. Show you understand risk management in distributed systems.*

Build your own story

Your 6 stories cover your strongest proof points — use this template whenever a new experience comes to mind before your interview.

Start with a real resume bullet or achievement. Don't start with a story idea — start with a fact. A metric, a deliverable, a result you can stand behind. Everything else builds from there.

✓ ANCHOR The real achievement — copy from your resume or write it in one sentence

STORY TITLE

COMPETENCY TAGS (PICK 1–2)

USE THIS STORY FOR — WHAT INTERVIEW QUESTIONS DOES IT ANSWER?

WHICH META VALUE DOES THIS BEST DEMONSTRATE?

Move fast | Focus on long-term impact | Build awesome things | Live in the future | Be direct and respect your colleagues | Meta, metamates, me

Circle one — be honest. If it's split between two, pick the one the story demonstrates most clearly.

SITUATION Draft

Set the context. What was the state of things before you acted? Keep to 2–3 sentences. Use [brackets] for anything you're not 100% certain about yet.

TASK Draft

What were you specifically responsible for? Why you, not someone else?

ACTION Draft

What did you specifically do? Name your decisions, not just activities. Every claim needs a "why I chose this" — that's where interviewers probe hardest.

RESULT ✓ Anchor here

Start with the metric from your anchor above — that's your verified fact. Then add business impact, recognition, or follow-on effects.

STRESS-TEST WITH AI

Once you've drafted your story, paste this into Claude or ChatGPT:

> *"Act as a Meta interviewer. I'm going to tell you a STAR story. After I finish, push back with 3 follow-up questions that test whether my answer is specific, credible, and genuinely demonstrates strong performance for this company. Be tough."*

Before you use this story

- [ ] Can you state the result metric from memory, without checking notes?
- [ ] In the Action, can you explain every decision and why you made it — not just what you did?
- [ ] Have you practiced this out loud at least once, timing it at 2–3 minutes?
- [ ] If the interviewer asks "what would you do differently?" — do you have an honest answer ready?

Interviewers won't ask the exact questions we prepared for — but if your story is solid, you can answer any version of the question. The goal isn't to memorise. It's to know the beats so you can deliver naturally.

6

What They're Testing — And How to Answer It

12 question patterns decoded — what's really being assessed, and how to answer any version of each question.

Note: Meta SWE onsites in 2026 include one AI-assisted coding round alongside one traditional coding round, one system design or product architecture round (infrastructure vs product role respectively), and one behavioral round. The AI-assisted round is 60 minutes in a CoderPad environment with an AI tool — code execution is OFF, using the AI is optional, and interviewers evaluate ownership and critical thinking, not AI proficiency. System design questions map to real Meta products — expect social-graph scale, News Feed architecture, or Messenger-style real-time systems.

COMPANY Derived from Meta interview structure

ROLE Standard for Software Engineer interviews

JD Derived from your job description

HOW TO USE THIS SECTION

These 12 questions were built specifically for your Meta SWE interview. The distribution — 2 coding / 4 behavioral / 3 system design / 1 ai assisted coding — reflects how Meta actually structures this interview at your level, based on their published evaluation criteria and hiring patterns.

Each question includes three layers of prep intelligence: what the interviewer is actually evaluating beneath the surface, what a strong answer demonstrates, and the patterns that cause candidates to fall short.

Work through every question before your interview. For behavioral questions, draft your answer using the Story Builder in Section 5 — then practice saying it out loud until the delivery feels natural.

1 Technical — System Design

ROLE Candidate's distributed systems experience

> **"At Stripe, you built an idempotency layer that reduced duplicate charge incidents by 94%. Walk me through your approach to ensuring idempotency in distributed systems. How would you apply those same principles to WhatsApp's message delivery infrastructure where we need to guarantee exactly-once delivery to 2 billion users?"**

WHAT THEY'RE REALLY ASKING

The interviewer wants to see if you can connect your real experience to Meta's scale challenges while demonstrating deep technical understanding of distributed systems guarantees. They're evaluating your ability to architect solutions for extreme scale (2B users) while maintaining correctness—a core competency for WhatsApp infrastructure engineers.

WHAT GREAT LOOKS LIKE

- Technical Depth: Explains idempotency keys, deduplication windows, and distributed consensus challenges from your Stripe experience with specific implementation details
- Scale Translation: Articulates how WhatsApp's 2B user scale changes the problem—sharding strategies, global distribution, storage constraints that weren't issues at Stripe's scale
- Delivery Guarantees: Discusses exactly-once vs at-least-once tradeoffs, client-side dedup, server-side validation, and how mobile connectivity affects reliability
- Real Implementation: References actual technologies (Kafka, distributed caches, vector clocks) and explains failure modes you've seen in production

RED FLAGS

- Generic Solutions: Gives textbook answers without connecting to your actual Stripe implementation or demonstrating real distributed systems experience
- Scale Blindness: Treats 2B users the same as smaller scale, missing the fundamental architectural differences and bottlenecks
- Correctness Gaps: Glosses over edge cases like network partitions, clock skew, or mobile client behavior that break idempotency guarantees
- Implementation Vagueness: Can't explain the specific technical choices you made at Stripe or how they'd need to change for messaging infrastructure

YOUR PREP

Use your Payment Idempotency System story as the foundation—be ready to explain your specific implementation (dedup keys, time windows, storage decisions) then extend it to messaging scale. Research WhatsApp's actual architecture challenges: client-server sync, offline message queuing, and cross-data center replication.

PRACTICE WITH AI

Act as a Meta interviewer asking about my Stripe idempotency experience. Push me on specific implementation details, then challenge me on how exactly-once delivery works differently for messaging vs payments at 2 billion user scale.

2 Coding — Algorithms

ROLE LeetCode medium with messaging context

> **"You're given an array of integers representing message send timestamps in milliseconds. Write a function that returns the maximum number of messages that can be sent within any 1-second sliding window, given that our rate limiter allows at most K messages per second per user."**

WHAT THEY'RE REALLY ASKING

This tests your fundamental algorithm skills with a sliding window problem disguised in a messaging context. The interviewer wants to see clean code, optimal time complexity (O(n)), and your ability to handle edge cases while thinking through rate limiting logic that's core to messaging infrastructure.

WHAT GREAT LOOKS LIKE

- Optimal Algorithm: Implements sliding window with two pointers or queue-based approach achieving O(n) time complexity
- Edge Case Handling: Considers empty arrays, single elements, K=0, and timestamps that are exactly 1000ms apart
- Clean Code: Writes readable, well-structured code with meaningful variable names and clear logic flow
- Rate Limiting Understanding: Explains how this connects to real-world rate limiting and discusses alternative approaches (token bucket, leaky bucket)

RED FLAGS

- Suboptimal Complexity: Uses nested loops or other $O(n^2)$ approaches instead of recognizing the sliding window pattern
- Bug-Prone Code: Makes off-by-one errors with the 1-second window calculation or doesn't handle boundary conditions properly
- Poor Communication: Jumps into coding without explaining approach or doesn't walk through test cases to verify correctness
- Missing Context: Treats this as pure algorithm exercise without connecting to practical rate limiting scenarios

YOUR PREP

This is a classic sliding window maximum problem—practice the two-pointer technique and queue-based approaches. Focus on the 1000ms window calculation (inclusive/exclusive boundaries) and think through how this applies to real rate limiting systems you've worked with.

PRACTICE WITH AI

Act as a Meta interviewer giving me this sliding window coding problem. Watch for optimal time complexity, edge cases, and clean code. Follow up by asking how this relates to real-world rate limiting systems.

3 Technical — System Design

COMPANY Meta's actual WhatsApp scale challenges

> **"Design WhatsApp's real-time messaging infrastructure to handle 2 billion users sending 100 billion messages per day. Focus on message routing, delivery guarantees, and handling offline users. How do you ensure sub-second delivery while maintaining exactly-once semantics?"**

From JD: *Design and build distributed backend systems for WhatsApp messaging infrastructure*

WHAT THEY'RE REALLY ASKING

This directly mirrors the infrastructure challenges this WhatsApp team faces daily. The interviewer wants to see if you can design production-grade systems at Meta's scale, understanding the specific constraints of mobile messaging: battery life, network reliability, global latency, and storage costs at 100B messages/day.

WHAT GREAT LOOKS LIKE

- Architecture Clarity: Proposes clear separation between connection management, message routing, delivery, and storage layers with specific technology choices
- Scale Calculations: Works through actual numbers—storage requirements, network bandwidth, database sharding strategies for 100B daily messages
- Mobile-First Design: Addresses push notifications, connection pooling, message compression, and offline sync specific to mobile messaging
- Delivery Guarantees: Explains exactly-once semantics using message IDs, acknowledgments, and deduplication without performance degradation

RED FLAGS

- Scale Naivety: Proposes solutions that work for thousands but break at billions—single database, simple load balancing, ignoring network partitions
- Generic Web Design: Applies standard web architecture without considering mobile constraints like battery, intermittent connectivity, or push notification limits
- Correctness Handwaving: Claims exactly-once delivery without explaining the distributed systems complexity or acknowledging CAP theorem tradeoffs
- Missing Operations: Ignores monitoring, rollback strategies, or how to deploy changes to infrastructure serving 2B users without downtime

YOUR PREP

Research WhatsApp's actual technical challenges: Erlang-based architecture, connection pooling strategies, message routing at global scale. Study how they handle offline users, cross-datacenter replication, and exactly-once delivery. Focus on the specific constraints of mobile messaging vs web applications.

PRACTICE WITH AI

Act as a Meta interviewer asking me to design WhatsApp's messaging infrastructure. Drill into specific scale calculations, mobile constraints, and exactly-once delivery mechanisms. Challenge me on how my design handles real failure scenarios.

4 Culture Fit — Cross-Team Influence

COMPANY Meta's cross-functional collaboration culture

> **"Tell me about a time when you had to make a critical technical decision that impacted multiple teams, but you didn't have complete buy-in initially. How did you drive alignment and what was the outcome?"**

WHAT THEY'RE REALLY ASKING

Meta's culture emphasizes direct communication and driving impact across organizational boundaries. The interviewer wants to see if you can influence without authority, navigate technical disagreements constructively, and build consensus around complex decisions—essential skills for senior engineers at Meta's scale.

WHAT GREAT LOOKS LIKE

- Stakeholder Mapping: Identified all affected parties, understood their concerns, and tailored communication to address specific team needs and constraints
- Data-Driven Persuasion: Used concrete metrics, prototypes, or pilot results to build evidence for your technical approach rather than relying on authority
- Collaborative Problem-Solving: Incorporated feedback to improve the solution, found win-win outcomes, and gave credit to others while driving the initiative forward
- Measurable Impact: Delivered concrete business or technical outcomes that validated the decision and strengthened relationships for future collaboration

RED FLAGS

- Authority Dependence: Relied on manager escalation or formal authority instead of building genuine technical and business consensus
- Poor Listening: Pushed through without addressing legitimate concerns or adapting the solution based on valid feedback from other teams
- Weak Follow-Through: Made the decision but failed to ensure successful implementation or didn't measure/communicate the positive outcomes
- Relationship Damage: Achieved short-term technical goals but left teams feeling steamrolled or excluded from the decision-making process

YOUR PREP

Draw from your Zero-Downtime Migration or Engineering Team Growth stories where you had to coordinate across teams at Stripe. Frame it using Meta's 'Be Direct and Respect Your Colleagues' principle—show how you were honest about tradeoffs while building genuine consensus through data and collaboration.

PRACTICE WITH AI

Act as a Meta interviewer asking about cross-team influence. Probe for specific stakeholder management tactics, how I built consensus without authority, and what the measurable outcomes were. Challenge me on difficult moments and relationship management.

5 Technical — AI-Assisted Coding

COMPANY Meta's new AI-assisted coding interview format

> **"I'm going to give you a coding problem and you'll solve it using CoderPad's AI assistant. Your task is to implement a distributed hash ring for consistent hashing. Here's the key: I want to see how you direct the AI, validate its suggestions, and ensure the final solution meets production requirements. Walk me through your thought process as you use the AI tool."**

From JD: *Engage with the AI-assisted coding tools Meta is rolling out across engineering*

WHAT THEY'RE REALLY ASKING

Meta is evaluating your ability to effectively collaborate with AI coding tools while maintaining engineering judgment. They want to see if you can direct AI assistance strategically, validate generated code critically, and ensure production-quality outcomes rather than blindly accepting AI suggestions.

WHAT GREAT LOOKS LIKE

- Clear Problem Breakdown: Start by explaining consistent hashing concepts and requirements before engaging the AI
- Strategic AI Direction: Give specific, well-reasoned prompts that guide the AI toward optimal solutions
- Critical Validation: Question AI suggestions, test edge cases, and explain why certain approaches won't work in production
- Production Mindset: Consider scalability, fault tolerance, and operational concerns that AI might miss

RED FLAGS

- Passive AI Usage: Simply accepting whatever the AI generates without critical evaluation or direction
- Missing Context: Failing to provide the AI with sufficient context about production requirements and constraints
- No Testing Mindset: Not considering edge cases, error handling, or how to validate the solution works correctly
- Over-reliance: Letting the AI drive the solution rather than using it as a sophisticated tool under your guidance

YOUR PREP

Practice the consistent hashing algorithm independently first so you can confidently guide and validate AI suggestions. Focus on developing a systematic approach: break down the problem, give specific prompts, critically evaluate responses, and always think about production concerns the AI might overlook.

PRACTICE WITH AI

Act as a Meta interviewer conducting an AI-assisted coding session. Give me a consistent hashing problem and evaluate how effectively I direct the AI, validate its suggestions, and ensure production readiness. Challenge me when I accept AI suggestions too quickly.

6 Culture Fit — Ownership

ROLE Candidate's specific technical ownership experience

> **"Your distributed rate-limiting service at Stripe handles 2M+ requests/second. Describe a specific production incident or performance bottleneck you encountered with this system. What went wrong, how did you debug it, and what was your role in the resolution?"**

WHAT THEY'RE REALLY ASKING

The interviewer wants to understand your ownership mentality and incident response skills at scale. They're evaluating whether you take true ownership of complex systems, how you handle pressure during outages, and your ability to drive systematic improvements rather than just quick fixes.

WHAT GREAT LOOKS LIKE

- Specific Technical Depth: Describe exact failure modes, metrics, and debugging techniques used on your rate-limiting system
- Ownership Throughout: Show how you drove the incident from detection through resolution and post-mortem improvements
- Systems Thinking: Demonstrate understanding of cascading effects and how rate-limiting failures impact downstream services
- Long-term Impact: Explain preventive measures and architectural improvements you implemented post-incident

RED FLAGS

- Vague Generalities: Speaking in abstract terms about incidents rather than providing specific technical details
- Passive Role: Describing what the team did rather than your personal contributions and decisions
- Quick Fix Mentality: Focusing only on immediate resolution without addressing root causes or prevention
- No Learning: Failing to demonstrate how the incident led to improved processes or system design

YOUR PREP

Use your Incident Response Optimization story but adapt it to focus specifically on a rate-limiting or high-throughput system failure. Frame it around the technical ownership you took, specific debugging steps, and systematic improvements you drove afterward to prevent recurrence.

PRACTICE WITH AI

Act as a Meta interviewer asking about production incidents with distributed systems. Probe deeply into my technical ownership, specific debugging approaches, and long-term improvements. Challenge me if I'm too vague about my personal contributions.

7 Coding — Algorithms

ROLE LeetCode medium tree traversal problem

> **"Given a binary tree where each node contains a character, write a function to find the longest path where all characters form a palindrome. The path can start and end at any nodes, not necessarily root to leaf."**

WHAT THEY'RE REALLY ASKING

This tests your ability to solve complex algorithmic problems involving tree traversal and string manipulation. Meta wants to see clean code structure, optimal time complexity understanding, and your approach to handling the palindrome constraint across tree paths.

WHAT GREAT LOOKS LIKE

- Clear Problem Analysis: Break down the problem into tree traversal + palindrome validation components
- Efficient Algorithm: Use DFS with path tracking and optimize palindrome checking (frequency counting vs string comparison)
- Edge Case Handling: Consider single nodes, empty trees, and paths with odd/even length palindromes
- Code Quality: Write clean, readable code with proper variable names and logical structure

RED FLAGS

- Brute Force Approach: Checking every possible path without optimizing for palindrome properties
- Incorrect Path Logic: Misunderstanding that paths can go through nodes (not just parent-child relationships)
- Poor Complexity: Using inefficient string operations instead of character frequency tracking
- Incomplete Solution: Missing edge cases or not handling the tree traversal correctly

YOUR PREP

Practice tree traversal problems (DFS/BFS) and palindrome detection separately first. Focus on a systematic approach: understand the problem constraints, design the algorithm step-by-step, consider time/space complexity, then implement with clean code structure.

PRACTICE WITH AI

Act as a Meta interviewer giving me tree traversal coding problems. Evaluate my problem-solving approach, algorithm efficiency, and code quality. Push me to optimize and handle edge cases thoroughly.

8 Technical — System Design

COMPANY Meta's actual Instagram infrastructure challenges

> **"Design Meta's content delivery system for Instagram Stories. 500 million users post Stories daily, each Story can be viewed by an average of 200 people, and Stories expire after 24 hours. How do you handle global distribution, caching strategies, and cleanup of expired content?"**

WHAT THEY'RE REALLY ASKING

Meta is testing your ability to think at their scale while understanding the unique constraints of ephemeral content. They want to see if you can design systems that handle massive throughput, optimize for short-lived data, and align with Meta's infrastructure philosophy of global distribution and user experience.

WHAT GREAT LOOKS LIKE

- Scale Awareness: Address 500M daily posts and 100B daily views with specific storage and bandwidth calculations
- Smart Caching Strategy: Design multi-tier caching (CDN, regional, local) optimized for 24-hour content lifecycle
- Efficient Cleanup: Implement automated expiration systems that don't impact serving performance
- Meta Integration: Consider how this fits with Meta's existing infrastructure (WhatsApp, Facebook, Instagram shared systems)

RED FLAGS

- Ignoring Scale: Designing for much smaller systems without acknowledging the massive throughput requirements
- Over-engineering: Adding unnecessary complexity instead of focusing on the core challenges of ephemeral content
- Poor Resource Management: Not optimizing for the 24-hour lifecycle or considering storage/bandwidth costs
- Generic Solutions: Using standard system design patterns without adapting to Meta's specific infrastructure needs

YOUR PREP

Research Meta's actual infrastructure challenges and their emphasis on efficiency at scale. Focus on how Meta's culture of 'Move Fast' applies to infrastructure decisions, and consider how this system would integrate with their existing multi-product ecosystem. Emphasize data efficiency and user experience optimization.

PRACTICE WITH AI

Act as a Meta interviewer asking about large-scale system design for ephemeral content. Evaluate my understanding of Meta's scale, infrastructure philosophy, and ability to optimize for short-lived data lifecycle. Challenge me on specific technical decisions.

9 Culture Fit — Execution

COMPANY Meta's 'Move Fast' cultural value

> **"Tell me about a time when you had to deliver a critical project with an aggressive timeline. How did you balance moving fast with maintaining code quality and system reliability?"**

WHAT THEY'RE REALLY ASKING

The interviewer is evaluating how you embody Meta's 'Move Fast' value while maintaining engineering rigor. They want to see if you can make smart tradeoffs under pressure, communicate risks effectively, and deliver quality outcomes when speed matters most.

WHAT GREAT LOOKS LIKE

- Clear tradeoff framework: Demonstrates a systematic approach to balancing speed vs. quality with specific examples of what you prioritized and deprioritized
- Proactive risk communication: Shows how you kept stakeholders informed about technical debt, shortcuts taken, and mitigation plans
- Iterative delivery strategy: Breaks down the project into phases with clear success criteria, allowing for fast feedback loops
- Quality guardrails: Maintains non-negotiable standards (security, core functionality) while being flexible on nice-to-haves

RED FLAGS

- No tradeoff strategy: Claims they maintained perfect quality while moving fast without explaining how or what sacrifices were made
- Blame shifting: Focuses on unreasonable timelines or management pressure rather than their own decision-making process
- Technical debt denial: Doesn't acknowledge shortcuts taken or fails to show learning about long-term consequences
- Hero complex: Presents solution as purely individual effort without showing collaboration or delegation skills

YOUR PREP

Use your Zero-Downtime Migration story - this shows exactly how you balanced speed with reliability when migrating critical payment systems. Focus on your phased approach, how you communicated risks to stakeholders, and specific quality guardrails you maintained despite time pressure.

PRACTICE WITH AI

Act as a Meta interviewer asking about balancing speed with quality. Probe for specific tradeoffs made, how I communicated risks, and what I learned about moving fast without breaking things.

10 Culture Fit — Growth Mindset

COMPANY Meta's learning from failure culture

> **"Describe a significant technical decision or project where you made a mistake or the outcome wasn't what you expected. What specifically did you do wrong, how did you handle it, and what did you learn?"**

WHAT THEY'RE REALLY ASKING

This probes your self-awareness, growth mindset, and ability to learn from failures - critical for Meta's fast-moving environment. They're looking for genuine vulnerability, specific lessons learned, and evidence that you've changed your approach based on past mistakes.

WHAT GREAT LOOKS LIKE

- Genuine ownership: Takes full responsibility for the mistake without deflecting blame or making excuses
- Specific failure details: Describes exactly what went wrong, what they missed, and why their approach was flawed
- Concrete learning application: Shows how the lesson changed their future decision-making with specific examples
- Process improvements: Implemented systemic changes to prevent similar failures, not just personal lessons

RED FLAGS

- Humble bragging: Presents a 'failure' that's actually a success or shows them in a positive light
- External blame: Attributes the failure to circumstances, other people, or lack of resources rather than their own decisions
- Vague lessons: Offers generic learnings like 'communicate better' without specific behavioral changes
- No follow-through: Can't demonstrate how the lesson was applied in subsequent situations

YOUR PREP

Think of a real technical failure from your background - perhaps an incident during the Payment Idempotency System development or a mistake in the SMS Delivery Optimization project. Be prepared to share specific details about what you did wrong, not just what went wrong around you.

PRACTICE WITH AI

Act as a Meta interviewer asking about a significant technical mistake I made. Push me to be more specific about my personal failures and probe whether I'm truly owning the mistake or deflecting responsibility.

11 Technical — System Design

JD Scaling candidate's experience to WhatsApp requirements

> **"At Stripe, you built a webhook delivery system with at-least-once guarantees and exponential backoff. If you were to build a similar notification system for WhatsApp that needs to handle push notifications for offline users across 2 billion accounts, how would you architect it differently given the scale difference?"**

From JD: *Design and build distributed backend systems for WhatsApp messaging infrastructure*

WHAT THEY'RE REALLY ASKING

The interviewer is testing your ability to scale architectural thinking from your proven experience to Meta's massive scale requirements. They want to see if you understand the fundamental differences in distributed systems at 2 billion users and can adapt your successful patterns accordingly.

WHAT GREAT LOOKS LIKE

- Scale-aware architecture: Recognizes that 2B users requires fundamentally different approaches - geographic distribution, edge computing, message queuing at unprecedented scale
- WhatsApp-specific constraints: Understands offline user challenges, mobile battery optimization, and global infrastructure requirements
- Proven pattern evolution: Takes successful webhook patterns and shows how exponential backoff, delivery guarantees, and retry logic must evolve
- Infrastructure trade-offs: Discusses consistency vs. availability, storage partitioning, and how to handle cross-region message delivery

RED FLAGS

- Linear scaling thinking: Simply suggests 'more servers' or 'bigger databases' without understanding distributed systems complexity
- Ignoring mobile constraints: Doesn't consider battery life, network connectivity, or offline scenarios critical for WhatsApp
- Over-engineering: Proposes complex solutions without justifying why they're needed at this scale
- Missing delivery semantics: Fails to address how at-least-once delivery guarantees change when dealing with billions of offline users

YOUR PREP

Start with your webhook delivery system from your background, then systematically think through how each component would need to change for 2B users. Focus on geographic distribution, offline user queuing, and how your exponential backoff strategy would need to evolve for global scale.

PRACTICE WITH AI

Act as a Meta interviewer asking about scaling notification systems to WhatsApp's scale. Challenge my assumptions about distributed systems and probe for specific technical decisions around offline user handling and global infrastructure.

12 Culture Fit — Leadership

JD Mentoring requirement in job description

> **"You're tasked with mentoring a junior engineer who's struggling to meet deadlines and their code quality needs improvement. The engineer is on your team and reports to the same manager. How would you approach this situation while maintaining team velocity and the engineer's confidence?"**

From JD: *Mentor junior engineers and contribute to engineering excellence*

WHAT THEY'RE REALLY ASKING

This evaluates your ability to embody Meta's collaborative culture while driving engineering excellence. They want to see if you can balance individual growth, team productivity, and maintain psychological safety - key aspects of Meta's 'Metamates' value.

WHAT GREAT LOOKS LIKE

- Structured mentorship approach: Creates clear development plan with specific skills gaps, measurable goals, and regular check-ins
- Velocity preservation: Balances immediate team needs with long-term engineer development through pair programming, code reviews, and task sizing
- Confidence building: Uses positive reinforcement, celebrates small wins, and provides constructive feedback that focuses on growth
- Cross-functional thinking: Considers impact on team dynamics, manager relationship, and ensures transparent communication

RED FLAGS

- Micromanagement: Suggests taking over the engineer's work or controlling their every move rather than teaching
- Confidence crushing: Focuses only on problems without providing constructive solutions or positive reinforcement
- Team neglect: Prioritizes individual mentoring without considering impact on overall team velocity and morale
- Manager bypassing: Doesn't involve or communicate with the shared manager about development progress and challenges

YOUR PREP

Draw from your Engineering Team Growth story where you mentored team members. Focus on Meta's 'Metamates' value - how you'd put the team member's growth alongside team success. Emphasize collaborative problem-solving rather than directive management.

PRACTICE WITH AI

Act as a Meta interviewer asking about mentoring struggling engineers. Probe for specific examples of how I balance individual development with team needs and test my understanding of Meta's collaborative culture.

7

Scripts for Awkward Questions

How to handle gaps, weaknesses, and curveballs with confidence.

Your Gap Analysis

JD REQUIREMENT	YOUR RESUME EVIDENCE	STATUS
Strong proficiency in Python, Go, C++, or Java	Strong professional experience with Python (Stripe payment processing pipeline, Twilio SMS delivery) and Go (distributed rate-limiting service, API gateway). However, no C++ experience shown for infrastructure role.	⚠ Gap
Experience designing and building distributed systems at scale	Extensive distributed systems experience: built idempotency layer for 1B+ monthly API calls, owned distributed rate-limiting service handling 2M+ requests/second, designed webhook delivery system, built SMS delivery pipeline for 500M+ messages/month	✓ Covered
3+ years of software engineering experience	5 years of software engineering experience (2018-Present) with progressive roles from Software Engineer to Senior Software Engineer	✓ Covered
Collaborate with cross-functional teams to define technical requirements and architecture	Resume shows mentoring engineers and conducting interviews, but lacks clear evidence of cross-functional collaboration with product, design, or other non-engineering teams	⚠ Gap
Engage with the AI-assisted coding tools Meta is rolling out across engineering	No mention of experience with AI coding tools, GitHub Copilot, or similar AI-assisted development tools	⚠ Gap

Bridge Scripts for Your Gaps

1 C++ Experience

Re: Strong proficiency in Python, Go, C++, or Java

WHY INTERVIEWERS WILL PROBE THIS

The role likely involves systems programming or performance-critical infrastructure where C++ is commonly used. Without C++ experience, the candidate might struggle with memory management, low-level optimizations, or integrating with existing C++ codebases.

YOUR BRIDGE SCRIPT

You're right that I don't have production C++ experience yet. My systems programming background has been primarily in Go and Python, where I've built high-performance distributed services handling millions of requests per second. I'm excited about the opportunity to learn C++ at [Meta/company name] *- I've been studying it independently through* [specific resource like Coursera, books, or projects] *and understand the fundamentals of memory management and performance optimization. Given my strong foundation in systems design and my track record of quickly mastering new technologies like* [example of tech you learned quickly]*, I'm confident I can become productive in C++ within* [realistic timeframe]*.*

BEFORE YOU USE THIS SCRIPT, VERIFY:

- [ ] What C++ learning resources have you actually started using?
- [ ] Can you give a specific example of quickly learning a new programming language or technology?
- [ ] What's a realistic timeframe for you to become productive in C++ given your current workload?

PRACTICE WITH AI

Act as a Meta infrastructure engineering interviewer. Challenge my answer about lacking C++ experience. Push back if my response sounds evasive or overly rehearsed, and ask follow-up questions about specific C++ concepts or learning progress.

2 Cross-functional Collaboration

Re: Collaborate with cross-functional teams to define technical requirements and architecture

WHY INTERVIEWERS WILL PROBE THIS

Meta's engineering roles require close partnership with product managers, designers, data scientists, and business stakeholders. Without demonstrated cross-functional experience, the candidate might struggle to translate business requirements into technical solutions or advocate for engineering constraints.

YOUR BRIDGE SCRIPT

While my resume focuses on the technical systems I've built, I've actually worked closely with cross-functional teams throughout my career. For example, when building [specific project like the idempotency layer] *, I collaborated extensively with* [product managers/business stakeholders] *to understand the business impact of duplicate charges and worked with* [customer success/support teams] *to define success metrics. I also partnered with* [data science/analytics teams] *to measure the 94% reduction in incidents. I'm excited about Meta's collaborative culture and bringing my experience translating business needs into scalable technical solutions.*

BEFORE YOU USE THIS SCRIPT, VERIFY:

- [ ] Can you identify specific non-engineering stakeholders you've worked with on your major projects?
- [ ] What business requirements or constraints influenced your technical decisions?
- [ ] How did you measure success beyond just technical metrics?

PRACTICE WITH AI

Act as a Meta interviewer asking about cross-functional collaboration. Challenge my answer by asking for specific examples of working with product managers or resolving conflicts between engineering and business priorities. Push back if my examples sound vague.

Bridge Scripts for Your Gaps

3 AI Coding Tools

Re: Engage with the AI-assisted coding tools Meta is rolling out across engineering

WHY INTERVIEWERS WILL PROBE THIS

Meta is investing heavily in AI-assisted development tools and expects engineers to leverage these for productivity gains. Without experience using tools like GitHub Copilot or similar AI assistants, the candidate might be slower to adopt Meta's AI coding initiatives or miss opportunities to accelerate development.

YOUR BRIDGE SCRIPT

I haven't had formal experience with AI coding tools at my previous companies, but I'm genuinely excited about this aspect of the role. I've been experimenting with [GitHub Copilot/ChatGPT for code/other AI tool] *on personal projects and I'm impressed by how it can accelerate* [specific use case like boilerplate generation, test writing, or debugging]*. Given my experience building developer productivity tools like* [API gateway/infrastructure tools you've built]*, I understand how the right tooling can transform engineering efficiency. I'm eager to learn about Meta's AI coding initiatives and contribute to their adoption across the team.*

BEFORE YOU USE THIS SCRIPT, VERIFY:

- [ ] Have you actually tried any AI coding tools, even informally?
- [ ] What specific developer productivity improvements have you worked on or observed?
- [ ] Can you articulate why AI-assisted coding matters beyond just 'writing code faster'?

PRACTICE WITH AI

Act as a Meta interviewer focused on AI coding tools adoption. Challenge my answer about lacking AI coding experience by asking about specific AI tools, how I'd measure their impact, or concerns about code quality. Push back if I sound like I'm just saying what you want to hear.

When You Don't Know the Answer

UNIVERSAL FRAMEWORK

The 4-Step Recovery

1. **Pause**
   2-3 seconds of silence is fine. Don't panic.
2. **Acknowledge**
   "That's a great question. I haven't encountered that exact scenario."
3. **Reason**
   "Here's how I'd think about it..."
4. **Anchor**
   "In a similar situation, I [relevant experience]..."

WHAT TO AVOID

- ✗ Bluffing or making up answers
- ✗ Saying "I have no idea" and stopping
- ✗ Getting visibly flustered
- ✗ Overexplaining why you don't know

PHRASES THAT WORK

"I haven't worked with that specific technology, but here's how I'd approach learning it..."

"That's outside my direct experience, but my instinct would be to..."

"I'd want to understand more about [X] before giving a definitive answer, but my initial thinking is..."

Curveball Questions

These questions are designed to test how you think under pressure. There's rarely a "right" answer — they're evaluating your reasoning process.

"Tell me about the biggest technical mistake you made at work. What happened and what did you change?"

Why they ask: Tests ownership and Focus on Long-Term Impact — Meta wants engineers who own failures completely, learn fast, and build systems that prevent recurrence. Deflection or blame is a strong negative signal.

FRAMEWORK

- → Name the mistake clearly and specifically — what failed, what was the impact on users or the system?
- → Own your role fully — no blame of tooling, teammates, or external factors
- → Walk through your diagnosis: how did you find it, what was the root cause?
- → Explain the permanent fix you implemented — not just the immediate mitigation
- → Show what changed in your process, monitoring, or design approach to prevent recurrence

⚠ Choose a real mistake with meaningful technical scope — not a minor bug. The quality of your reflection and what changed in your process matters more than the size of the error.

"Design Instagram's News Feed ranking system. How would you decide what content to show each user?"

Why they ask: Tests system design at Meta's core social-network scale — ranking, personalisation, real-time constraints, and the tension between engagement signals and long-term user value. A flagship Meta system design question.

FRAMEWORK

- → Start with requirements: what does good mean for the user and for Meta? Engagement vs satisfaction vs retention?
- → Propose a two-stage architecture: candidate generation (retrieve N posts from social graph) → ranking (score and order)
- → Walk through ranking signals: social proximity, content type affinity, recency, predicted engagement probability
- → Address the real-time constraint: how do you serve a ranked feed in <200ms at 2B+ users?
- → Discuss A/B testing and how you would measure whether a ranking change improved long-term user value, not just short-term clicks

⚠ Study two-stage ranking architectures (candidate generation + ranking). Think about signals (social graph, content type, recency, engagement history), freshness vs personalisation tradeoffs, and how you would A/B test ranking changes.

Quick Reference: The Graceful Bridge

For any gap or weakness, remember: **Acknowledge → Pivot → Evidence**. Never deny a gap exists. Instead, show adjacent experience and genuine enthusiasm to grow. Interviewers expect gaps — they're evaluating how you handle them, not whether you're perfect.

8

Questions That Make Them Want You

Strategic questions to ask each interviewer type.

The questions you ask tell interviewers as much about you as your answers. Great questions demonstrate that you've **done your research**, you're **thinking strategically** about the role, and you're **evaluating them**, not just hoping to be chosen.

Use **2-3 questions per interview** — more than that feels like an interrogation. Choose based on who you're talking to.

For the Recruiter

Focus: Process, timeline, culture fit

"What does the interview process look like from here, and what's a realistic timeline?"

Why it works: Shows you're organized and serious about moving forward.

"What traits have you seen in candidates who really thrive here?"

Why it works: Gets insider perspective on culture fit — recruiters have pattern-matched hundreds of hires.

★ COMPANY-SPECIFIC

"In your experience screening candidates for WhatsApp Infrastructure roles, what distinguishes candidates who successfully navigate the systems design interviews from those who struggle?"

Why it works: This recruiter sees many candidates go through the technical screens and onsite systems design interviews. They can share patterns of what makes candidates successful at Meta's specific interview format, which is valuable for this candidate's preparation.

For the Hiring Manager

Focus: Role expectations, success metrics, team dynamics

"If I were crushing it in this role after 6 months, what would that look like?"

Why it works: Shows you're thinking about impact, not just tasks. Reveals their real priorities.

★ COMPANY-SPECIFIC

"How does 'Move fast' actually play out when you're making infrastructure decisions that affect 2 billion WhatsApp users? I'm curious about the balance between speed and the reliability requirements at this scale."

Why it works: This hiring manager can speak to how the 'Move fast' leadership principle is lived day-to-day on their specific team, especially given the tension between speed and reliability in infrastructure work.

◆ ROLE-SPECIFIC

"You mentioned 'own features end-to-end from design through production deployment and monitoring' - coming from payments infrastructure where I owned similar full-stack ownership, what does that end-to-end ownership look like specifically for WhatsApp's messaging systems?"

Why it works: Connects the candidate's payments infrastructure experience at Stripe to the specific end-to-end ownership requirement in this role. The HM can detail what this responsibility actually entails for their team.

For Peer Interviewers

Focus: Day-to-day reality, collaboration, culture

"What do you wish you'd known before joining?"

Why it works: Invites honest perspective and shows you value candor.

★ COMPANY-SPECIFIC

"I've read about Meta's flat organizational structure requiring constant cross-team collaboration. Day-to-day, how do you actually get things done when you need something from another team but can't just tell them what to do?"

Why it works: This peer lives the reality of Meta's flat structure daily and can give candid insight into how 'leadership through influence' actually works in practice, beyond the cultural description.

◆ ROLE-SPECIFIC

"What's it like collaborating with the cross-functional teams mentioned in the role? I'm thinking about how infrastructure decisions get made when you're working with product, security, or other engineering teams."

Why it works: This peer can share the real experience of the cross-functional collaboration that's core to this role, from someone who does this work daily rather than from a hiring perspective.

For Executives / Skip-Level

Focus: Company direction, strategic importance, vision

"Where do you see this product/team in 2-3 years?"

Why it works: Shows you're thinking long-term and want to understand the strategic trajectory.

★ COMPANY-SPECIFIC

"Given Meta's big bets on AI and the metaverse, how does investing in WhatsApp's messaging infrastructure fit into the company's long-term strategy? I'm curious about the 'Focus on long-term impact' lens here."

Why it works: An executive can speak to the strategic importance of WhatsApp infrastructure within Meta's broader platform and AI investments, connecting to the 'Focus on long-term impact' principle.

◆ ROLE-SPECIFIC

"What drove the decision to prioritize this WhatsApp Infrastructure engineering hire right now? I'm interested in understanding how this role connects to Meta's platform strategy."

Why it works: An executive can explain the strategic reasoning behind prioritizing this infrastructure role and how it fits Meta's broader technical and business objectives.

Questions That Hurt Your Candidacy

Avoid asking: "What does your company do?" (shows no research) • "How soon can I get promoted?" (sounds entitled) • "What's the work-life balance like?" (ask instead: "How does the team approach deadlines?") • "Did I get the job?" (awkward) • Anything easily Googleable (shows no prep) • "I don't have any questions" (signals disinterest)

MAKE THESE YOUR OWN

The personalized questions above are based on your target company and role.

- → Don't read these verbatim — internalize the intent and ask in your own words
- → Reference recent news or product launches you've seen
- → Mention something from the interviewer's LinkedIn (if visible)
- → Connect your specific experience to their challenges
- → If you know someone who works there, ask what they'd want to know

9

A Handout That Closes the Deal

Your 30/60/90 day approach — tailored to Meta and your background.

WHY THIS WORKS

Show How You Think, Not What You'll Deliver

A 30/60/90 day plan signals **strategic thinking** and **self-awareness**. But the best plans don't over-promise — they show how you'll approach the role based on your specific background, while leaving room for the reality that you don't yet know what it's like to work there.

We've tailored this plan to your experience and Meta's expectations. Print the next page and bring it to your interview — or offer to send it afterward.

Your Printable Handout

The next page is a clean, one-page summary designed to leave with your interviewer.
It has your name on it — no Interview101 branding — so it looks like you created it.

→ Next Page

My 30/60/90 Day Approach

Marcus Webb
SWE

DAYS 1-30 — Ramp

Build the foundation to contribute effectively

WHAT I'LL SEEK TO UNDERSTAND

- WhatsApp messaging infrastructure scale requirements and distributed system architecture patterns
- Cross-team collaboration processes and technical decision-making frameworks at Meta
- Move Fast culture: shipping before feeling ready and breaking things boldly

WHERE MY BACKGROUND HELPS

- Stripe payments experience translates to WhatsApp scale: 1B+ API calls to 2B+ users
- Production reliability expertise from on-call ownership and MTTR reduction applies directly

MY MILESTONE

Ship a real fix and shadow deployments while mapping key messaging infrastructure metrics

DAYS 31-60 — Ship

Add value while still learning

WHERE I'LL LOOK TO ADD VALUE

- Apply distributed systems expertise to messaging reliability challenges
- Leverage API design experience for WhatsApp infrastructure improvements
- Support urgent infrastructure needs using production debugging skills

WHAT I'M EXCITED TO LEARN

- Excited to master Meta's AI-assisted coding tools and social platform engineering patterns

MY MILESTONE

Owned feature end-to-end from design doc to production with A/B test results

DAYS 61-90 — Own

Begin driving, not just supporting

WHERE I MIGHT ADD UNIQUE VALUE

- Begin leading distributed system design decisions for messaging infrastructure components
- Start owning reliability improvements using production incident response experience
- Take initiative mentoring team members through complex technical problems

WHAT I'LL DIAGNOSE FIRST

- What are the highest-impact tech debt areas in current messaging infrastructure?
- Where can system design improvements deliver measurable reliability gains?
- What technical architecture decisions does my manager see as highest priority?

MY MILESTONE

Proposed architecture for cross-team technical decision with data-backed system improvement plan

---

This is my approach, not a rigid commitment — I'll adapt based on what I learn.

interview101 **Interview Day Cheat Sheet**

Marcus Webb
SWE at Meta

YOUR 30-SECOND PITCH

Senior engineer with 5 years building distributed systems at scale. Designed payment idempotency layer reducing duplicates by 94% across 1B+ monthly calls. Ready to ship fast on infrastructure that connects billions at Meta.

YOUR STORY BANK — READY TO USE

1. **Payment Idempotency System**
   → Tell me about a time you built something impactful
2. **Zero-Downtime Migration**
   → Tell me about a time you balanced speed with reliability
3. **Incident Response Optimization**
   → Tell me about a time you improved a process
4. **Engineering Team Growth**
   → Tell me about a time you mentored someone
5. **API Gateway Reliability**
   → Tell me about a time you built something reliable
6. **SMS Delivery Optimization**
   → Tell me about a time you solved a complex problem

KNOW ABOUT META

- → **Values:** Move Fast, Focus on Long-Term Impact, Intellectual Honesty
- → **Recent:** AI-assisted coding now standard; prepare CoderPad with execution OFF
- → **Interview:** Assess algorithms, data structures, and systems design thinking
- → **Culture:** Own features end-to-end; drive cross-functional alignment through influence

YOUR TOP SELLING POINTS

- ✓ Large-scale messaging infrastructure at billions of requests
- ✓ End-to-end ownership from architecture to production monitoring
- ✓ Technical leadership through mentorship and code review influence
- ✓ Shipped production systems handling 2M+ RPS with sub-3ms latency

IF THEY ASK ABOUT DATA STRUCTURES AND SYSTEMS DESIGN DEPTH

My strengths are in applied systems at scale, not algorithmic foundations. I'm closing this gap with focused study on core data structures and design patterns. My infrastructure work required deep optimization thinking—I'm translating that rigor into structured problem-solving.

CURVEBALL READY

"Tell me about your biggest technical mistake at work"

Name mistake clearly, own it fully (no blame). Walk through root cause diagnosis. Explain permanent fix and how you changed monitoring or process. Connect to Focus on Long-Term Impact—show you prevented recurrence.

QUESTIONS TO ASK

HIRING MANAGER
"How does Move Fast actually play out when infrastructure decisions affect 2B WhatsApp users? What's the balance between speed and reliability at this scale?"

PEER
"How do you actually get things done day-to-day in a flat structure when you need something from another team but can't direct them?"

EXECUTIVE
"How does investing in WhatsApp messaging infrastructure fit Meta's long-term strategy around AI and the metaverse?"

RECRUITER
"What distinguishes candidates who excel in systems design interviews from those who struggle?"

⚡ REMEMBER

- → Coding rounds: get to a working solution fast, then optimise — interviewers are timing your path to correctness, not waiting for the perfect answer
- → Code execution is OFF in CoderPad — practise writing and mentally tracing code without running it
- → AI-assisted round: treat the AI as a junior engineer — direct it clearly, review its output critically, own everything it produces; using it is optional
- → Behavioral carries leveling weight — E4 vs E5 is often decided by the depth of ownership and cross-team impact in your stories