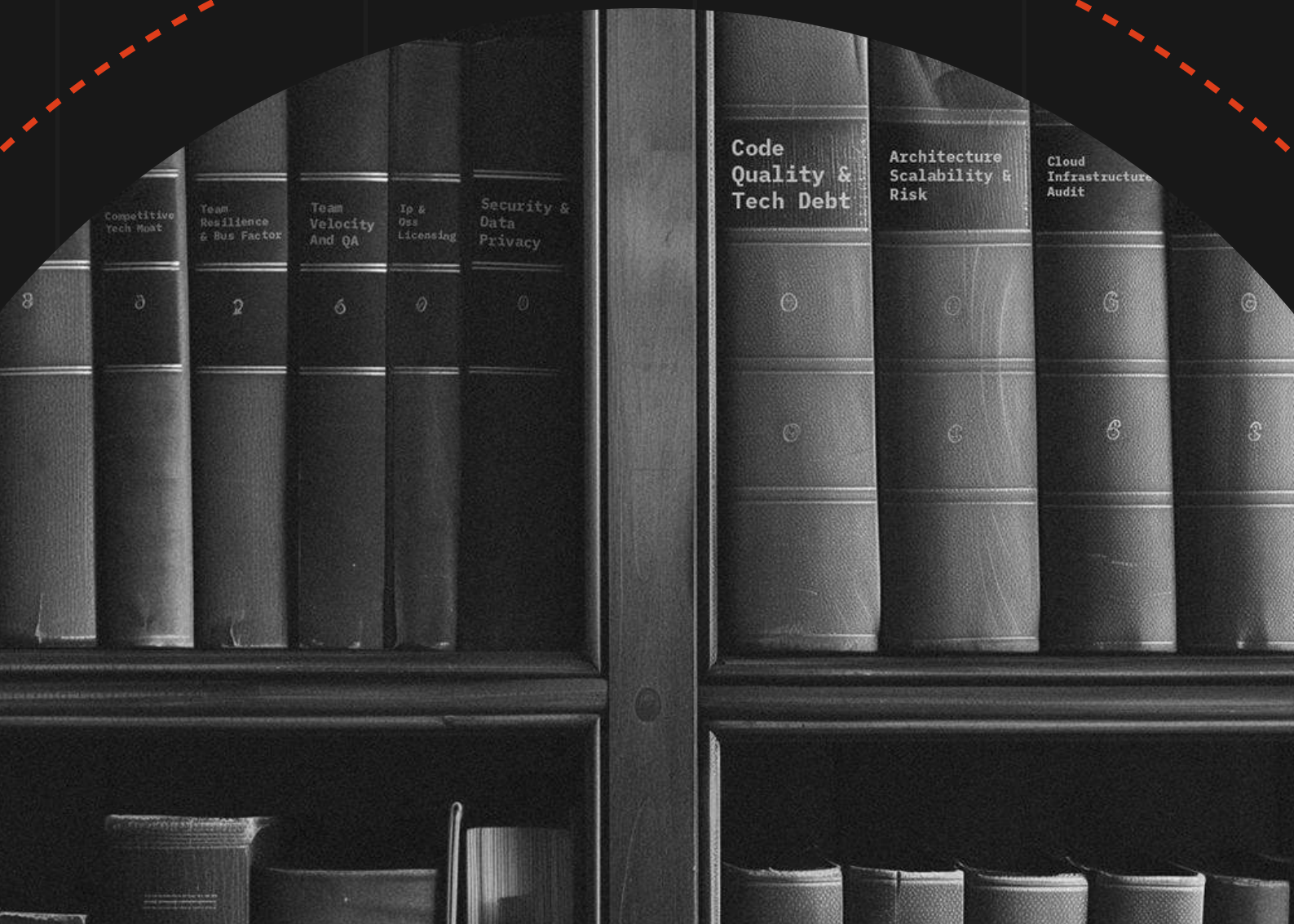


LLM PROMPT LIBRARY FOR TECH AUDITS









Contents

Takeaway	3
Design Principles for an Audit Prompt	4
Prompt Library Structure (Examples)	6
Code Quality & Tech Debt	6
Architecture Scalability & Risk	8
Cloud Infrastructure Audit	11
Security & Data Privacy	13
Ip & Oss Licensing	16
Team Resilience & Bus Factor	18
Team Velocity And Qa	20
Infra Cost & Unit Economics	23
Competitive Tech Moat	25
M&A Integration Readiness	28
Risk Matrix & Remediation Plan	30
About the Author	33

Takeaway

Curating a reusable prompt library transforms a general-purpose LLM from a clever chatbot into a structured, audit-grade assistant. With well-written, role-specific prompts, teams can:

-  **Cut prep time**
-  **Reduce errors and inconsistencies in documentation**
-  **Identify high-risk issues before external auditors or buyers do**
-  **Standardize internal reviews across engagements**

Prompt libraries function as reusable assets—living audit playbooks that can be improved and versioned over time.



Design Principles for an Audit Prompt

Principle	Strategic Impact	How to Implement
1. Role Priming	Sets the “expert voice” of the LLM, ensuring insights match investor expectations.	Start every prompt with: “You are a senior [domain] auditor...” (e.g., “You are a senior cloud security auditor preparing a sell-side report.”). Tailor to SaaS, FinTech, AI, etc.
2. Context Block	Aligns analysis with the company’s real-world deal, tech stack, and risk profile.	Include: company stage (Series B/C or Exit), domain (e.g., B2B SaaS), revenue size, tech stack (e.g., Node.js on GCP), known red flags, investor type (PE vs VC).
3. Task & Output Specification	Produces investor-usable results like risk matrices, technical summaries, and remediation plans.	Ask for structured formats like: <ul style="list-style-type: none"> • Risk Table (Issue, Severity, File/Domain, Recommendation) • Cost-to-Remediate Estimates • Traffic-Light Summary (Green/Yellow/Red by domain)
4. Dual-Track Evaluation	Balances growth potential (investor upside) with operational readiness (integration/exit risk).	Use this weighting per scenario: <ul style="list-style-type: none"> • Series B: 70% Growth / 30% Ops • Series C: 60% / 40% • M&A/Exit: 40% / 60%. Ensure prompts trigger both types of assessment.
5. Risk-First Orientation	Identifies red flags proactively—before buyers do—enhancing trust and reducing post-LOI renegotiations.	Instruct AI to flag top 5 risks per domain. Format: Risk – Likelihood – Impact – Mitigation. Focus on investor-sensitive categories: security gaps, IP, dependencies, team bus factor, legal exposure.
6. Document/ Data Room Readiness	Accelerates deal closure by generating data-room ready assets.	Add “Generate folder structure and filename list for data room” or “List required artifacts per diligence domain.” Include naming conventions.
7. Stage-Tailored Depth	Matches prompt depth to the company’s maturity and deal context—avoiding too shallow or too technical outputs.	For Series B: Ask for growth risk, technical scaling blockers, and roadmap friction. For Series C/Exit: Demand operational KPIs, documentation checks, process readiness, legal exposure, integration friction.

Design Principles for an Audit Prompt

Principle	Strategic Impact	How to Implement
8. Stakeholder-Specific Notes	Gives founders and execs tailored insights: one cares about valuation and legacy; the other about continuity and team dynamics.	Output sectioned by audience: <ul style="list-style-type: none"> • Founders: Wealth protection, risk to brand/exit value, co-founder transitions • Execs: Post-deal role clarity, continuity risks, talent lock-in plans
9. Constraints & Guardrails	Prevents hallucination and limits generic or irrelevant content.	Use lines like: “Cite only industry standards,” “Avoid generic advice,” or “Use only provided file references.”
10. Self-Check or Rubric	Promotes LLM quality control and investor-grade outputs.	Add: “Review your output for relevance, investor clarity, and format adherence. Flag any gaps in source data.” Include scoring prompts for confidence (e.g., “Rate architecture risk on a 1–5 scale”).
11. Smart Field Thinking	Encourages modular, reusable outputs—mirroring how tools like Kira and Litera tag due diligence fields.	Break outputs by “smart fields”: e.g., ‘Security Misconfigurations,’ ‘IP Ownership,’ ‘Unit Economics Red Flags.’ This makes outputs easier to reuse in different decks, rooms, or narratives.
12. Human-in-the-Loop Validation	Ensures LLM outputs are auditable, defensible, and ready for expert validation.	After generation, prompt AI to suggest verification steps. Example: “Which areas require manual confirmation?” or “Flag any output with low source certainty.”
13. Embedded Best Practices	Raises audit bar and avoids missed signals.	In each domain, insert “Evaluate against [relevant standard],” e.g., OWASP Top 10, AWS Well-Architected, SOC 2, ISO 27001. Provide known benchmarks if possible (e.g., “Typical Series B code coverage: 60–70%”).

Prompt Library Structure (Examples)

Replace bracketed variables with engagement specifics; keep or adapt bullets.

CODE QUALITY & TECH DEBT



Situation:

- **{deal_stage}** (e.g., Series B, Series C, or Exit)
- Impacts the lens of analysis:
 - **Series B** → Growth and scalability risks
 - **Series C** → Documentation, test coverage, team/process maturity
 - **Exit** → Integration readiness, tech debt visibility, valuation blockers



Role: Senior Software Due Diligence Engineer



Task: Evaluate the codebase for maintainability, technical debt, and audit-readiness based on the deal context



Prompts

1. Context & Intent Initialization

"You are a senior software due diligence auditor. The target company is currently at the {deal_stage} stage. The codebase is a monorepo written in {language}, ~{LOC} lines of code, built with {frameworks}. Your task is to evaluate maintainability, audit readiness, and code health in a way that informs {deal_stage}-specific investors (e.g., VC, PE, acquirer). Prioritize risks that would impact valuation, integration, or development velocity."

2. Code Quality Assessment

"Rate the overall code quality (1–10) and justify your score. Identify major issues such as code smells, large or tightly coupled classes, poor naming, and violations of modularity. Provide file paths or line references if possible."

3. Test Coverage Evaluation

"Review the test coverage report and identify modules with low coverage (e.g., <60%). Emphasize high-risk areas that lack sufficient tests, especially those affecting business-critical logic."

4. Refactor Prioritization

"List the top 5 components that would benefit from refactoring. Include estimated developer effort (in dev-days) and a rationale for each choice, especially how it relates to {deal_stage} concerns (e.g., future growth, documentation debt, integration blockers)."

5. Cost-to-Replicate Estimation

"Estimate the cost to rebuild the entire codebase using a COCOMO II-based model. State your assumptions about team size, hourly rate, and productivity."

6. Stage-Specific Risk Translation

"Identify 3–5 risks that would be most concerning to investors at the {deal_stage}. These may include lack of documentation, unclear ownership, brittle modules, or gaps in test/process hygiene."

7. Human Validation Suggestions

"Propose 3 follow-up questions that a human reviewer should ask the engineering team to validate unclear areas, assess team ownership, or flag technical surprises before close."



Required Artifacts:

- Monorepo or zipped code sample
- Test coverage report (HTML or JSON)
- Static analysis output (e.g., SonarQube)
- Contributor notes, README, style guides



Security Tips:

- Remove all credentials (.env, .pem, .ssh, etc.)
- Strip usernames, emails, and PII from commits
- Share via time-bound, read-only access links only



Output Review & Debugging Tips:

- Are referenced file paths real and verifiable?
- Is the cost estimate tailored (not generic)?
- Are “top risks” aligned with {deal_stage} logic (e.g., integration for Exit)?
- Are refactor suggestions actionable and not vague?

[ARCHITECTURE SCALABILITY & RISK]



Situation:

- {deal_stage} (e.g., Series B, Series C, Exit)
- Key priorities shift by stage:
 - **Series B** → Detect scaling bottlenecks and immature architecture choices
 - **Series C** → Assess process hardening, documentation, and automation gaps
 - **Exit** → Focus on system stability, integration risks, and technical liabilities



Role: Principal Architect



Task: Assess the scalability, resilience, and integration readiness of the current architecture



Prompts

1. Setup & Strategic Context

"You are a principal software architect conducting an audit of a cloud-native SaaS product hosted on {cloud_provider}. The company is at the {deal_stage} stage. Your task is to assess the scalability, resilience, and operational risk of the system. Prioritize findings that would affect {deal_stage}-relevant concerns, such as investor confidence, integration cost, or platform stability."

2. Architecture Snapshot Generation

"Summarize the current high-level system architecture. Identify the main components, how they communicate, and where core services are hosted. If possible, provide a component map grouped by domain/function."

3. Bottleneck Identification

"Analyze the architecture for potential bottlenecks or weak points. Highlight any single points of failure, tightly coupled components, or services that may not scale well under 5x traffic. Use a Bottleneck Matrix format: Component / Risk / Trigger / Impact / Mitigation."

4. Cloud-Native Best Practices Check

"Evaluate the system's adherence to cloud-native and distributed systems best practices (e.g., autoscaling, retries, statelessness, observability). Flag any technical debt or configuration drift that might create risk at {deal_stage} scale or during a post-acquisition transition."

5. AWS Well-Architected Alignment (if applicable)

"Review architecture through the lens of the AWS Well-Architected Framework (or applicable cloud framework). For each pillar (e.g., Reliability, Performance Efficiency, Operational Excellence), rate alignment on a scale of 1–5 and note high-risk misconfigurations."

6. Integration Readiness Assessment

"What parts of the system would likely create friction during M&A integration? For example, hard-coded assumptions, tight internal dependencies, or undocumented interfaces. Suggest mitigation or refactor actions."

7. Critical Follow-Up Questions

"Suggest 3–5 follow-up questions an engineering leader should ask the team to clarify ownership, assumptions, or roadmap unknowns in critical components."



Required Artifacts:

- High-level architecture diagram (PDF or draw.io)
- Component descriptions or service map
- System documentation (e.g., Confluence export)
- Performance dashboards or scaling benchmarks (Grafana, Datadog)



Security Tips:

- Strip internal IPs, secrets, and environment-specific configs
- Anonymize auth/data workflows, especially PII flowcharts
- Do not include acquirer-specific system overlays or integrations



Output Review & Debugging Tips:

- Are bottlenecks explained clearly with realistic load conditions?

- Are system diagrams accurate and complete (vs oversimplified)?
- Do integration risks reflect the current **{deal_stage}** lens?
- Are Well-Architected ratings consistent and supported by examples?

[CLOUD INFRASTRUCTURE AUDIT]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Focus areas shift by stage:
 - **Series B** → Rapid deployment risk, immature IaC, weak guardrails
 - **Series C** → Infrastructure reliability, CI/CD maturity, cost-efficiency
 - **Exit** → Cloud spend optimization, IAM exposures, audit trail gaps



Role: DevOps Auditor



Task: Identify misconfigurations, cost inefficiencies, and security risks in the cloud stack



Prompts

1. Stage-Aware Audit Framing

"You are a senior DevOps auditor reviewing the infrastructure of a SaaS company preparing for a {deal_stage} round. The environment is managed via Terraform and deployed on {cloud_provider}. CI/CD is handled with GitHub Actions. Your task is to flag configuration risks, IAM gaps, and pipeline maturity issues that could impact valuation, reliability, or security posture."

2. IAM Risk Assessment

"Analyze the provided IAM configuration (e.g., Terraform, IAM policy files).

Identify overly permissive roles, wildcard actions, or missing separation of duties. Rate each finding by severity and remediation difficulty."

3. CI/CD Pipeline Maturity

"Evaluate the CI/CD pipeline YAMLs for completeness, security, and quality gates. Are there missing checks (e.g., static analysis, secret scanning)? Is there support for rollback, artifact signing, and staged deployments? Score overall CI/CD maturity on a 1–10 scale."

4. Infrastructure Resilience Review

"Review the infrastructure-as-code setup. Flag misconfigurations such as unencrypted S3 buckets, public-facing RDS, or missing availability zone redundancy. Provide findings in table format: Resource / Issue / Severity / Suggested Fix."

5. Cost Optimization Readiness

"Based on infra usage reports and cloud spend data, are there areas of overprovisioning or unused resources? Suggest 3 quick wins for reducing cost without harming performance or scale."

6. {deal_stage}-Specific Risk Surface

"Highlight infrastructure risks most relevant to {deal_stage} investors. Focus on auditability, scaling risk, and post-close maintainability (e.g., single-region deployments, untracked resource drift, hard-coded configs)."

7. Follow-Up Questions for DevOps Team

"Suggest 3 questions an auditor should ask the DevOps team to confirm findings, clarify intent, or assess deployment reliability in high-traffic scenarios."



Required Artifacts:

- Terraform files, Helm charts, CloudFormation templates
- CI/CD YAMLS (e.g., [.github/workflows/*.yaml](#))
- Cost reports (AWS/GCP billing)
- Monitoring configs (Datadog, Prometheus)



Security Tips:

- Mask secrets in YAMLS and config files
- Strip IP addresses, access tokens, and ARNs
- Don't include [.pem](#) or [.ssh](#) keys in shared folders



Output Review & Debugging Tips:

- Are misconfigurations realistic, not generic?
- Do IAM findings include exact resource paths or policies?
- Is CI/CD maturity scored with a rationale?
- Are cost-saving tips supported by actual usage evidence?

[SECURITY & DATA PRIVACY]



Situation:

- [{deal_stage}](#) (e.g., Series B, Series C, Exit)
- Emphasis shifts by stage:
 - **Series B** → Look for immature security practices, lack of policy enforcement
 - **Series C** → Evaluate privacy compliance, breach history, and DSR readiness
 - **Exit** → Focus on unresolved vulnerabilities, audit trails, legal exposure, and acquirer liability



Role: Security Compliance Analyst



Task: Identify technical and procedural risks related to application security and data privacy



Prompts (Reflecting **{deal_stage}**)

1. Context Initialization with Risk Lens

"You are a security and compliance analyst auditing a SaaS product that processes {data_types} and is subject to {compliance_frameworks}. The company is preparing for a {deal_stage} event. Your goal is to identify security vulnerabilities and privacy compliance gaps that would raise red flags for legal, investor, or acquirer stakeholders."

2. OWASP & Infra Security Review

"Review the system's current security posture against OWASP Top 10. Flag any known vulnerabilities in authentication, session management, input validation, or data storage. Prioritize based on exploitability and impact."

3. Privacy Compliance Scan

"Evaluate the provided privacy policies, DSR workflows, and data flow diagrams. Identify any missing GDPR/CCPA elements such as consent logging, right to erasure, or cross-border data handling. Rate each issue as High, Medium, or Low risk."

4. Vulnerability History & Resolution Check

"Using the latest penetration test and vulnerability scans, summarize unresolved issues, including CVEs or OWASP categories. Indicate how long each has remained unpatched and suggest remediation timelines."

5. Data Exposure Mapping

"Map the flow of personally identifiable information (PII) and sensitive

data across the system. Highlight areas where data is stored, transferred, or exposed without encryption or proper controls. Output a simplified data flow with risk notes per step."

6. {deal_stage}-Relevant Risks for Legal/Buyers

"Highlight the 3–5 most investor-sensitive issues related to security or data privacy. Focus on unresolved vulnerabilities, weak access controls, legacy data exposure, or audit trail gaps that could delay the deal or increase acquirer liability."

7. Validation Qs for Security/Legal Team

"List 3 follow-up questions an auditor should ask the security or legal team to validate risk areas, such as unlogged access, lack of breach notification procedures, or unclear data retention rules."



Required Artifacts:

- Penetration test results
- Vulnerability scan reports (e.g., Qualys, Nessus)
- Data flow diagrams
- Consent logs, DSR request records, privacy policies
- SOC 2, ISO 27001, or GDPR audit artifacts (if available)



Security Tips:

- Redact usernames, email addresses, access tokens
- Remove unresolved high-severity vulnerabilities if public
- Never share raw CVE scan dumps in investor-facing docs



Output Review & Debugging Tips:

- Do issues map clearly to OWASP or regulatory categories?
- Are risks ranked by impact and exploitability?

- Does data flow mapping align with known system architecture?
- Are findings actionable, not just boilerplate?

[IP & OSS LICENSING]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Shifting focus by stage:
 - **Series B** → Early signs of license misuse, lack of IP assignment from freelancers
 - **Series C** → OSS license risks, incomplete IP documentation, IP control readiness
 - **Exit** → Legal exposure from copyleft licenses, missing contractor agreements, or ownership ambiguity



Role: IP & Licensing Reviewer



Task: Identify IP ownership risks and open-source license exposure that could impact acquirer risk or valuation



Prompts

1. Context Initialization with Risk Framing

"You are an IP and open-source licensing reviewer auditing a software company preparing for a {deal_stage} event. The stack is OSS-heavy with contributions from multiple contractors. Your goal is to identify any IP ownership gaps and licensing risks that could delay a deal or lead to legal exposure."

2. License Risk Scan

*"Review the provided package manifests (**package.json**, **pom.xml**, etc.)*

and list all open-source licenses used. Highlight any problematic licenses (e.g., GPL, AGPL, LGPL, or other copyleft variants). Rate license risk as High/Medium/Low."

3. IP Ownership Analysis

"Assess the documentation of IP ownership for contributions made by current and former employees, contractors, or third parties. Flag any missing IP assignment agreements or ambiguous contributor arrangements."

4. Contract Clause Review

"Review provided employment and contractor agreements. Extract and summarize clauses related to IP transfer, license grants, and confidentiality. Highlight any that are missing, one-sided, or unenforceable."

5. Buyer-Sensitive Red Flag Summary

"Highlight 3–5 IP or OSS license issues that would raise concern during {deal_stage} due diligence. Focus on acquirer-sensitive risks such as viral licenses, dual-licensing strategies, and lack of contributor documentation."

6. Human Follow-Up Guidance

"Suggest 3 follow-up questions a legal reviewer or CTO should ask the company to verify IP control and license compliance."



Required Artifacts:

- Dependency files: **package.json**, **requirements.txt**, **pom.xml**
- IP assignment agreements (employees & contractors)
- License scan reports (e.g., FOSSA, WhiteSource, ScanCode)

- Employment and contractor contracts with IP clauses
- Git logs or commit history with contributor identities



Security Tips:

- Strip PII from contracts (e.g., names, emails) before review
- Share license reports only — not raw source code
- Redact any internal contributor notes tied to unpublished IP



Output Review & Debugging Tips:

- Are risky licenses flagged with correct legal implications?
- Are contributor/IP gaps clearly tied to real docs or missing docs?
- Is there a clear risk table showing potential deal-blockers?
- Are all findings linked to verifiable sources (e.g., filenames, scan reports)?

[TEAM RESILIENCE & BUS FACTOR]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Focus shifts by stage:
 - **Series B** → Key-person risk, undocumented tribal knowledge, unbalanced engineering
 - **Series C** → Module ownership, onboarding difficulty, contributor churn
 - **Exit** → Continuity risk, team attrition exposure, knowledge silos, talent lock-in readiness



Role: Team Dependency Analyst



Task: Map team structure, ownership coverage, and resilience to personnel loss or transition



Prompts

1. Context Setup with Stage-Specific Goals

"You are a team dependency and resilience auditor. The company is approaching a {deal_stage} event. The engineering team has {team_size} members, working across a stack that includes {tech_stack}. Your goal is to assess knowledge silos, contributor risk, and overall team resilience to transitions, exits, or integration."

2. Ownership Mapping Prompt

"Using git commit data, map which contributors own which modules or code areas. Highlight any modules owned solely by one person or with minimal recent activity."

3. Bus Factor Assessment

"Generate a 'Module × Person' matrix. Identify any single points of failure where only one engineer understands or maintains a critical area. Suggest mitigation actions (e.g., pairing, documentation, code walkthroughs)."

4. Tenure and Churn Review

"Assess the team's tenure distribution. Flag risks from short-tenure contributors owning key systems, or recent turnover in core areas."

5. {deal_stage}-Specific Risk Framing

"From an investor or acquirer perspective at the {deal_stage}, which people-related risks matter most? Focus on bus factor, onboarding difficulty, contributor availability, or risk of team attrition post-close."

6. Follow-Up Question Suggestions

"Suggest 3–5 questions to ask the engineering lead or HR team to confirm module ownership, plan for knowledge transfer, or talent retention risk."



Required Artifacts:

- Git commit history and contributor stats
- Engineering org chart or team roster
- HR tenure reports (anonymized)
- Documentation ownership logs or team wiki export
- System/module documentation, if available



Security Tips:

- Use role labels instead of names (e.g., “Senior Backend Dev”)
- Never share personal emails or direct HR records
- Anonymize commit data and replace usernames with roles if possible



Output Review & Debugging Tips:

- Are module maps clearly linked to contributor IDs or roles?
- Is the bus factor framed in business impact terms, not just tech terms?
- Are risks realistic, not theoretical (e.g., tied to specific systems)?
- Do follow-up questions reveal gaps in institutional knowledge?

[TEAM VELOCITY AND QA]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Focus evolves by stage:
 - **Series B** → Assess team execution pace, early signs of velocity drag
 - **Series C** → Look for quality stability, process discipline, and release hygiene
 - **Exit** → Identify post-deal velocity risks, QA automation debt, brittle deployment cycles



Role: Engineering Performance Analyst



Task: Evaluate development throughput, QA maturity, and release process stability



Prompts

1. Context Setup with Stage-Specific Goals

"You are an engineering performance analyst. The company is at the {deal_stage} stage and follows a {methodology} process, releasing every {sprint_duration}. Your goal is to evaluate QA rigor, engineering velocity, and process maturity for investor and M&A readiness."

2. Test Coverage Trend Analysis

"Review the test coverage data. Are there coverage trends over time (increasing, decreasing, plateauing)? Identify areas with persistently low coverage or missing test ownership. Output a graph/table of coverage by module or timeframe."

3. Bug Backlog Assessment

"Analyze the current bug backlog. What percentage are high-severity or regressions? Are bugs clustered around specific modules or releases? Highlight patterns that might suggest poor testing or release instability."

4. Velocity Metrics Calculation

"Using sprint reports and JIRA exports, calculate average story throughput, lead time, and cycle time. Identify outliers (e.g., sprints with velocity drops or unresolved carryover)."

5. QA Process Maturity Evaluation

"Evaluate QA practices: Is there a defined testing strategy (unit,

integration, E2E)? Are test failures triaged and tracked? Score QA maturity (1–10) with justification per practice area."

6. {deal_stage}-Specific Framing for Buyers

"For a {deal_stage}-level investor or acquirer, which process risks would be most concerning? Focus on release predictability, test automation gaps, and potential scale challenges."

7. Follow-Up Questions for QA/Eng Leaders

"Suggest 3–5 questions for QA or engineering leads to clarify test coverage ownership, release blockers, and recent process changes."



Required Artifacts:

- QA test coverage reports (e.g., Jest, Mocha, Cypress)
- Bug backlog and severity reports
- Sprint burndown or velocity charts
- JIRA exports (stories, epics, bugs)
- QA run logs or test dashboards (e.g., Allure, TestRail)



Security Tips:

- Strip any ticket content referencing production incidents or sensitive users
- Anonymize JIRA user data in exports
- Remove internal system or product names not disclosed to investors



Output Review & Debugging Tips:

- Are velocity metrics calculated from real data, not assumptions?
- Do QA maturity scores map to observable artifacts?
- Are bug patterns explained by module, not just totals?

- Do prompts align with the pressures of {deal_stage} (e.g., faster releases, buyer scrutiny)?

[INFRA COST & UNIT ECONOMICS]



Situation:

- {deal_stage} (e.g., Series B, Series C, Exit)
- Financial focus shifts by stage:
 - **Series B** → Spot overbuild and future scaling risks
 - **Series C** → Scrutinize infra efficiency and ROI of infrastructure choices
 - **Exit** → Forecast growth impact, optimize cost-to-serve, justify spend to acquirers



Role: Technology Finance Analyst



Task: Decompose infrastructure spend, analyze unit economics, and model scaling costs



Prompts (Tied to {deal_stage} insights)

1. Context Initialization with Growth Lens

"You are a technology finance analyst reviewing cloud infrastructure spend and usage metrics for a company at the {deal_stage} stage. The company currently spends \${monthly_spend} per month and generates {ARR} in annual revenue, with a user base of {user_count}. Your goal is to break down infra cost drivers, assess unit economics, and model 10x growth scenarios."

2. Infra Spend Breakdown

"Decompose the total cloud spend into major categories (e.g., compute,

storage, bandwidth, monitoring, vendor tools). Flag any categories with outsized costs or unclear ROI."

3. Unit Economics Evaluation

"Calculate unit economics metrics: infra cost per active user, per feature use, and per dollar of ARR. Identify inefficient feature clusters or disproportionately expensive workflows."

4. Scaling Forecast Model

"Project infrastructure cost under a 10x user load. Assume proportional usage unless data suggests bottlenecks. Highlight what infra components will need right-sizing, auto-scaling, or re-architecture."

5. {deal_stage}-Relevant Cost Risks

"Highlight financial risks relevant to {deal_stage} stakeholders. Focus on hidden opex growth, third-party lock-in, or cost unpredictability under scale."

6. Benchmarking and Spend Reasonableness

"Compare current cost structure to similar companies in {industry}. Flag any red flags where the company overpays vs. industry norms (e.g., \$/MAU, \$/GB stored, \$/transaction)."

7. Follow-Up for CFO/CTO

"Suggest 3–5 questions that should be asked of finance or engineering leadership to confirm assumptions, check cost controls, or validate cloud budgeting practices."



Required Artifacts:

- Cloud billing reports (AWS/GCP/Azure exports)
- Infra spend dashboards (e.g., CloudZero, Cloudability, FinOps tools)

- Usage metrics by customer, feature, or endpoint
- Historical infra spend vs. growth logs
- Vendor invoices for third-party SaaS tools



Security Tips:

- Remove customer IDs from usage files
- Redact resource tags or project names that may expose roadmap
- Share aggregated spend only — avoid PII or raw logs unless anonymized



Output Review & Debugging Tips:

- Are scaling forecasts tied to real infra usage patterns?
- Are \$/user or \$/ARR ratios benchmarked with logic?
- Are major spend buckets tied to features, not just services?
- Do prompts consider acquirer concerns (cost to maintain or migrate)?

[COMPETITIVE TECH MOAT]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Focus evolves by stage:
 - **Series B** → Highlight unique innovation, roadmap defensibility, and velocity advantage
 - **Series C** → Emphasize scale, IP defensibility, and market differentiation
 - **Exit** → Show strategic value to acquirers (e.g., integration leverage, cost-to-rebuild advantage)



Role: Strategic Technology Advisor



Task: Position the company's technology as a competitive asset that enhances valuation and deal appeal



Prompts (Stage-aware and investor-facing)

1. Strategic Context Setup

"You are a strategic technology advisor helping a company articulate its tech advantage in preparation for a {deal_stage} investment or acquisition. The product operates in the {industry} sector and claims technical advantages in {domain}. Your job is to evaluate and position the tech stack, IP, and architecture as part of a broader valuation narrative."

2. Tech Moat Assessment

"Identify and explain the core technical differentiators. These may include proprietary algorithms, unique performance benchmarks, custom infrastructure, or time-to-build advantages. Output a Tech Moat Table: Advantage / Description / Evidence / Rebuild Difficulty (1–5)."

3. IP & Defensibility Summary

"Summarize any IP assets (e.g., patents, trade secrets, proprietary datasets) that contribute to defensibility. Highlight how hard these would be to replicate or reverse-engineer, and whether ownership is clearly assigned."

4. Valuation Angle for {deal_stage}

"Translate the tech moat into valuation value for a {deal_stage} investor. Explain whether it justifies premium pricing, cost leverage, faster roadmap, or acquirer synergy."

5. Comparative Landscape Analysis

"Compare the company's architecture or performance to top 3

competitors. Use benchmarks, third-party tests, or known design tradeoffs. Output: Area / This Company / Competitor A / Gap / Strategic Implication."

6. Strategic Fit Framing (for Exit deals)

"For M&A or Exit scenarios, explain how the company's tech aligns with a buyer's strategy. Could it speed integration? Provide unique distribution leverage? Replace legacy platforms? Include high-level buyer-fit notes."

7. Follow-Up Qs for Tech Execs or Board

"List 3–5 questions an investor or buyer would likely ask to validate the moat claims — e.g., about IP enforcement, roadmap defensibility, dependency on open-source components, or speed-to-market edge."



Required Artifacts:

- Tech comparison charts or benchmarks
- IP portfolio overview (patents, data assets, proprietary tools)
- Performance metrics or internal dashboards
- Analyst notes or buyer feedback on differentiation
- Competitor teardown or landscape analysis (if available)



Security Tips:

- Strip out detailed customer configs or proprietary implementation docs
- Redact unpublished IP details unless under NDA
- Avoid disclosing private roadmap commitments tied to specific buyers



Output Review & Debugging Tips:

- Are moat claims specific, verifiable, and framed in business value terms?

- Do competitive comparisons reflect actual gaps, not just vague claims?
- Is the valuation framing realistic and tied to known investor priorities?
- Are advantages clearly mapped to IP, time-to-build, or buyer-fit value?

[M&A INTEGRATION READINESS]



Situation:

- **{deal_stage}** (typically Exit, but also relevant at late-stage Series C)
- Stage-specific focus:
 - **Series C** → Identify integration blockers early and flag systems that may require rework
 - **Exit** → Prioritize post-close friction, undocumented dependencies, and acquirer risk



Role: M&A Tech Integrator



Task: Assess post-deal integration complexity and system compatibility



Prompts (Targeting **{deal_stage}**-aligned concerns)

1. Stage-Driven Setup with Buyer Lens

"You are a technical integrator assessing a company's engineering systems ahead of a {deal_stage} transaction. The infrastructure is based on {stack}, and the company uses {tools/platforms}. Your goal is to identify technical, architectural, and operational risks that could delay or complicate post-close integration with a buyer's environment."

2. System Compatibility Matrix

"List all major internal systems (e.g., auth, billing, analytics, data pipelines) and assess their compatibility with common enterprise platforms (e.g., Okta, Workday, Salesforce, Snowflake). Score each integration on a 1–5 scale and flag custom or hardcoded implementations."

3. Interface & Dependency Risk Review

"Analyze the provided API and interface documentation. Identify undocumented or tightly coupled services, legacy protocols, or fragile endpoints that may require decoupling. Output a Dependency Risk Table: Component / Risk / Impact / Integration Fix."

4. Infrastructure Compatibility Check

"Review infrastructure layers (e.g., cloud provider, CI/CD tooling, containerization approach) for compatibility with standard enterprise IT. Highlight any lock-in or unusual tooling that could block replatforming or integration."

5. Refactoring & Integration Effort Estimate

"Estimate the dev time and complexity (Low/Medium/High) required to make each system integration-ready. Include authentication, data syncing, and admin panel alignment."

6. {deal_stage}-Specific Integration Friction Risks

"Highlight the top 3–5 risks most likely to create timeline overruns or cost escalations during integration. Tailor to {deal_stage} investor concerns (e.g., slow replatforming, lack of audit trails, compliance gaps)."

7. Follow-Up Validation for Post-Close Owners

"List 3–5 questions to ask the engineering or IT operations team to

confirm readiness for integration or carve-out (e.g., identity federation support, data export tooling, version compatibility)."



Required Artifacts:

- Interface/API documentation and schemas
- Integration architecture diagrams
- ETL/data sync process maps
- System dependency matrix
- Versioning maps, tech stack registry, internal SSO/IAM setup



Security Tips:

- Remove roadmap references tied to a specific acquirer
- Strip proprietary or third-party integration details unless consented
- Redact internal org mappings or user directories if shared



Output Review & Debugging Tips:

- Is system compatibility assessed with realistic buyer stacks?
- Are interface risks tied to specific endpoints or services?
- Do refactor estimates reflect the actual coupling level?
- Are risks ranked by impact on post-close execution?

[RISK MATRIX & REMEDIATION PLAN]



Situation:

- **{deal_stage}** (e.g., Series B, Series C, Exit)
- Focus changes with stage:
 - **Series B** → Highlight growth blockers and immature processes
 - **Series C** → Balance scale friction with technical risk
 - **Exit** → Emphasize deal-stopping red flags, integration cost, and timeline threats



Role: Risk Manager



Task: Consolidate findings across audit domains into an executive-grade risk summary and remediation roadmap



Prompts (Output built for **{deal_stage}** diligence decks)

1. Setup & Output Intent

"You are a technical risk manager preparing a diligence summary for a company entering a {deal_stage} transaction window. Your task is to synthesize findings from prior audits (code, infra, team, security, integration) into a prioritized risk matrix and remediation plan suitable for investor or acquirer review."

2. Cross-Domain Risk Matrix Generation

"List the top risks by domain (e.g., code quality, architecture, security, team, compliance). For each: include Likelihood (High/Med/Low), Impact (High/Med/Low), and a short summary. Output in table format."

3. Severity-Based Risk Prioritization

"Rank the risks by combined severity. Highlight any red flags that would typically trigger a delay, re-scoping, valuation discount, or buyer pushback."

4. Remediation Plan Per Risk

"For each risk, suggest a practical mitigation or remediation action, including estimated time and role/team required. Output as: Risk / Fix / Owner / ETA / Status."

5. {deal_stage}-Specific Risk Translation

"Which risks are most sensitive at the {deal_stage} level? Frame them in

investor terms: integration friction, liability exposure, missed roadmap targets, or technical debt affecting M&A execution."

6. Board-Ready Summary Format

"Rewrite the risk matrix as a 1-page summary for a board or diligence committee. Use a traffic-light summary (Green/Yellow/Red per domain) and bullet points for top critical items."

7. Low-Confidence Area Flagging

"List any domains or audit areas where risk confidence is Low due to missing data, incomplete documentation, or lack of access. Suggest how to validate these post-close or in follow-up sessions."



Required Artifacts:

- Consolidated findings from prior audits
- Individual audit reports (Security, Code, Infra, Team, Licensing)
- Existing remediation playbooks or issue trackers
- Board decks or diligence report templates (if available)



Security Tips:

- Mark draft vs. confirmed risks clearly
- Redact findings not yet addressed in public or shared documents
- Avoid listing speculative risks without source evidence



Output Review & Debugging Tips:

- Are risks aligned to {deal_stage} investor concerns?
- Are mitigation steps specific, not vague ("improve coverage" → "add 20 tests in Module X")?
- Does the matrix format allow for easy board or exec consumption?
- Are confidence levels flagged where inputs were partial or missing?

About the Author

At MEV, we help businesses win with the right technology. That doesn't just mean writing good code — it means understanding what's at stake, asking important questions early, and solving problems before they turn into blockers.

For nearly 20 years, we've supported startups and growing companies through product launches, scale-up moments, infrastructure changes, and — more than a few — last-minute scrambles before investor or partner reviews. We've seen how much rides on technical decisions, and how often those decisions are made without the full picture.

That's why resources like this prompt library resonate with us. It reflects how we think and work: dig deep, get clear, and pressure-test assumptions before they become liabilities. We believe software should be built with context — business goals, user needs, tech constraints, and what's coming next — all on the table.

As part of that mindset, we offer **pre-deal technical audit and optimization services** for teams preparing for due diligence, exit, or internal readiness checks. We bring product, engineering, and infrastructure experience together to give you a clean, honest view of your system — and help you move forward with confidence.



+1 212-933-9921



solutions@mev.com



mev.com



linkedin.com