Technical Due Diligence Checklist

112 checks, from real assessments across seed stage through pre-acquisition. Built for investors, acquirers, board members and operators who need more than the founder's architecture slide.

You won't get through all of it in one session, and you shouldn't try. Run the smell test first. If it goes badly, the rest is documentation of a decision you have already made.

Before you start: what to ask for

Ask for these on day one. How fast they arrive, and how complete they are, is itself a finding. A team that can't produce a dependency list in a week has told you something.

OK	CON	N/A	
☐	☐	☐	Read access to the repositories, including history, not a zip of the tip
☐	☐	☐	An architecture diagram drawn by an engineer, not by marketing
☐	☐	☐	The dependency manifest, or better, a generated SBOM
☐	☐	☐	Last 12 months of cloud bills, unredacted
☐	☐	☐	Incident log or postmortems for the last year
☐	☐	☐	The on-call rotation and who is actually on it
☐	☐	☐	Any penetration test or security audit, including the ones that went badly
☐	☐	☐	SOC 2, ISO 27001 or equivalent report if they claim one
☐	☐	☐	IP assignment agreements for every employee and contractor who wrote code
☐	☐	☐	Open source policy and the last scan output
☐	☐	☐	Model cards and training data provenance, if any ML is in the product

Start here: the five-minute smell test

> Quick indicators before diving deep. Any "no" warrants deeper investigation. These five questions reveal more about technical health than hours of code review.

OK	CON	N/A	
☐	☐	☐	Can the CTO explain the architecture in plain English to a non-technical stakeholder?
☐	☐	☐	Is there code older than six months? (Complete rewrites suggest thrashing)
☐	☐	☐	Do they deploy at least weekly? (Monthly means cautious, or terrified of the codebase)
☐	☐	☐	Can they clearly describe their last outage and what changed after it?
☐	☐	☐	Do they know their deployment rollback time?

Architecture assessment

SYSTEM DESIGN

OK	CON	N/A	
☐	☐	☐	Architecture matches company stage (monolith for early, services only if needed)
☐	☐	☐	Can explain why they chose their architecture, not just “Netflix does it”
☐	☐	☐	Service boundaries are clean and logical, if microservices
☐	☐	☐	No premature optimization for scale they haven’t reached
☐	☐	☐	Clear data flow documentation exists

DATABASE AND DATA

OK	CON	N/A	
☐	☐	☐	Boring, proven database choices (Postgres usually wins)
☐	☐	☐	Each specialized database solves a real problem, not a resume-driven one
☐	☐	☐	Data backup and recovery process documented and tested
☐	☐	☐	No custom database or data layer, unless they’re a database company
☐	☐	☐	Schema migrations are version-controlled and reversible

DEPENDENCIES

OK	CON	N/A	
☐	☐	☐	Reasonable number of third-party dependencies
☐	☐	☐	Critical dependencies are from established vendors
☐	☐	☐	Fallback plan exists if a key vendor disappears or raises prices tenfold
☐	☐	☐	No obscure libraries for core business logic
☐	☐	☐	Dependencies are regularly updated (check the last update date)

AI and generated code

The 2026 addition, and the one most checklists haven’t caught up with. The question isn’t whether they used AI. Everyone used AI. The question is whether anyone understood what it produced.

OK	CON	N/A	
☐	☐	☐	Roughly what share of the codebase was AI-generated?
☐	☐	☐	Which tools, and are those tools still in the workflow?
☐	☐	☐	Who reviewed generated code before it merged, and is that review recorded?
☐	☐	☐	Pick a generated module. Can someone on the team explain it line by line?
☐	☐	☐	Can they debug it at 2am, or only regenerate it?
			—This is the real risk. Not bad code, unowned code.
☐	☐	☐	Are there tests around the generated parts, written by a human?

- [ ] [ ] [ ] Any AI-generated code carrying an incompatible or unknown licence?
- [ ] [ ] [ ] If the product calls a model at runtime: what happens when that provider has an outage, deprecates the model, or triples the price?
- [ ] [ ] [ ] Are prompts, model versions and parameters under version control?
- [ ] [ ] [ ] Is there a documented fallback when the model returns nonsense?

IF THEY TRAIN OR FINE-TUNE MODELS

OK CON N/A

- [ ] [ ] [ ] Where did the training data come from, and can they prove they had the right to use it?

RED FLAG

—Training data rights now show up as representations in purchase agreements. Unverified provenance is an assumed liability that scales with the dataset.

- [ ] [ ] [ ] Model cards and a training data datasheet exist, with known biases and limitations
- [ ] [ ] [ ] Model artifacts are versioned and reproducible from a recorded pipeline
- [ ] [ ] [ ] Any regulatory exposure identified, EU AI Act or sector rules

Code quality

TESTING

OK CON N/A

- [ ] [ ] [ ] Critical paths have test coverage
- [ ] [ ] [ ] Integration tests exist and run
- [ ] [ ] [ ] CI pipeline runs tests on every commit
- [ ] [ ] [ ] Tests actually fail when the code breaks

—Ask them to break something on a call and watch what happens

DOCUMENTATION

OK CON N/A

- [ ] [ ] [ ] README explains how to run it locally, and it works
- [ ] [ ] [ ] Architecture decisions are recorded
- [ ] [ ] [ ] API documentation exists
- [ ] [ ] [ ] Documentation is current, not two years stale

CODE HEALTH

OK CON N/A

- [ ] [ ] [ ] Git history shows healthy contribution patterns
- [ ] [ ] [ ] No single file that changes on every commit (usually broken)
- [ ] [ ] [ ] Recent refactoring has a clear rationale
- [ ] [ ] [ ] Team can identify their top three technical debt areas

Licensing and IP

The category that kills deals quietly. Copyleft contamination can force disclosure of proprietary source, which removes most of what you were buying. Nobody discovers this in month nine and feels good about it.

OK	CON	N/A	
☐	☐	☐	Does the company maintain a written open source policy?
☐	☐	☐	Do developers get any training on licence categories and obligations?
☐	☐	☐	Is there an approval process before a copyleft-licensed component goes in?
☐	☐	☐	Is open source usage tracked in an inventory, not reconstructed on request?
☐	☐	☐	Are scans run regularly, or only when an acquirer asks?
☐	☐	☐	Has any GPL or AGPL code been linked into the proprietary codebase? RED FLAG
			—This is the one that changes the price
☐	☐	☐	Does the company contribute code to outside projects, and who approves it?
☐	☐	☐	Is contributed code covered by a policy that protects confidential material?
☐	☐	☐	Do all employees have signed IP assignment agreements? RED FLAG
☐	☐	☐	Do all contractors, including the ones from five years ago? RED FLAG
☐	☐	☐	Any code written before incorporation, and who owns it?
☐	☐	☐	Are trademarks and domains held by the company rather than a founder personally?

Infrastructure and operations

DISASTER READINESS

OK	CON	N/A	
☐	☐	☐	Clear answer for “what if the primary database goes down?”
☐	☐	☐	Documented recovery time for complete data loss
☐	☐	☐	Deployment rollback tested and timed, not theorized
☐	☐	☐	Credential rotation process exists
☐	☐	☐	Incident response runbook documented

CLOUD AND COSTS

OK	CON	N/A	
☐	☐	☐	Cloud spend is proportional to usage and revenue
☐	☐	☐	Cost per user or per transaction is known
☐	☐	☐	No obvious waste: idle resources, over-provisioning, forgotten environments
☐	☐	☐	Growth in cloud costs won’t outpace revenue
☐	☐	☐	Committed spend agreements disclosed, with their end dates

Team and process

OK	CON	N/A	
☐	☐	☐	Bus factor greater than one for every critical system RED FLAG
☐	☐	☐	Knowledge is documented, not just in heads
☐	☐	☐	New engineer productive within two to four weeks
☐	☐	☐	Team acknowledges technical debt (denial is the red flag, not the debt)
☐	☐	☐	Clear process for responding to a security vulnerability
☐	☐	☐	Nobody critical is three months from vesting out

Security basics

OK	CON	N/A	
☐	☐	☐	Secrets in environment variables or a manager, not in the repo RED FLAG
☐	☐	☐	Check the git history too, not just the tip
☐	☐	☐	Authentication and authorization logic exists and is centralized
☐	☐	☐	No known-vulnerable versions of critical dependencies
☐	☐	☐	Someone has done at least one security review
☐	☐	☐	Credential rotation is possible if something is compromised
☐	☐	☐	Access controls exist and not everyone has admin
☐	☐	☐	Customer data is encrypted at rest and in transit RED FLAG
☐	☐	☐	There’s a written answer to “what happens when we get breached”

Red flags and yellow flags

Knowing the difference is most of the job.

Red. These kill deals or reprice them.

OK	CON	N/A	
☐	☐	☐	Fundamental scaling limitations that require a complete rewrite
☐	☐	☐	Security disasters: plaintext passwords, public buckets with customer data
☐	☐	☐	Copyleft contamination of the proprietary codebase
☐	☐	☐	Missing IP assignments from anyone who wrote material code
☐	☐	☐	Key person dependency with no mitigation plan
☐	☐	☐	Misrepresentation: claims that don’t match what you found
☐	☐	☐	Dangerous vendor lock-in on unfavourable terms
☐	☐	☐	Training data with no provable rights behind it

Yellow. These are negotiation points, not exits.

OK	CON	N/A	
☐	☐	☐	Technical debt exists and the team knows where it is
☐	☐	☐	Missing tests, fixable with time
☐	☐	☐	Junior team, which sets a mentorship expectation
☐	☐	☐	Unusual technology choices, which may be innovative or may be a problem
☐	☐	☐	Heavy AI-assisted development where the team still understands the output

Final assessment

OK	CON	N/A	
☐	☐	☐	Can this technology support the business plan as written?
☐	☐	☐	What could go wrong technically, and how likely is it?
☐	☐	☐	Is this team capable of building what they're proposing?
☐	☐	☐	What should be addressed in the first 90 days after the money lands?
☐	☐	☐	What would you need to see to change your answer?

> **Before the full review:** For a live deal, CodeProvenance.ai is a fixed-fee preflight ($25,000 to $40,000) that covers many of the code checks on this list. The target runs the scanner on its own machine, no source code leaves it, and I write a five-page memo 48 to 72 hours after the bundle arrives.