Technical due diligence checklist

Seven areas, and the checks an investor's assessor works through under each.

What it is

Technical due diligence is an investor's assessment of whether the engineering can carry the plan the money is buying. What comes out is not a grade. It is a list of findings, and findings get priced.

How to use it

Tick what you can evidence today. The rest is your preparation backlog, and most people find theirs in the last two areas rather than the first. Start two to three months before you open a round: the artefacts take longer to assemble than the fixes take to make.

01 Architecture and scalability

The question Whether the architecture was chosen or accumulated | **Evidence** Decision records with rejected alternatives

- [ ] The system's shape can be drawn on one page by someone who did not build it
- [ ] Every major choice has a written reason, including the ones you would make differently now
- [ ] You can name what breaks first at ten times current load, and roughly what it costs to fix
- [ ] Tenant or customer isolation is a rule enforced in code, not a convention people follow
- [ ] A new engineer can run the whole thing locally in under a day
- [ ] Nothing in production depends on a machine, an account or a service only one person can reach

02 Code quality and delivery

The question Whether the team can change the system safely and quickly | **Evidence** Tests that fail for the right reasons, and deploy history

- [ ] Tests exist that fail when you break the behaviour they name
- [ ] The time from merge to production is a number you know
- [ ] There is a rollback, and somebody has used it this quarter
- [ ] A new engineer's first change reached production in their first week, and you can name when that last happened
- [ ] Dependency versions are current enough that upgrading is a task rather than a project
- [ ] The build is green on the main branch right now, and if it is not, everybody knows why

03 Team and key-person risk

The question Whether the company's knowledge exists outside one head | **Evidence** Written conventions, and a second person who can deploy

- [ ] More than one person can deploy, and the second one has done it recently
- [ ] Each major subsystem has a named owner and at least one other person who has changed it
- [ ] Engineering conventions are written down somewhere a new starter gets pointed at
- [ ] Contractors have signed IP assignments, and you can produce them
- [ ] Someone other than the CTO can answer an architecture question in a call
- [ ] Holiday happens without a freeze

04 Security and compliance posture

The question Whether security is a habit or a document | **Evidence** Access controls, an audit trail, restore evidence

- [ ] You can list who can reach production data, and every access leaves a record
- [ ] No credential lives in the repository, and rotating one is a documented task
- [ ] Someone owns dependency vulnerabilities, with a decision rule for what gets patched when
- [ ] A backup has been restored, on purpose, and the restore was timed
- [ ] Every subprocessor touching customer data has a data processing agreement on file
- [ ] Customer-facing security questions have a written answer that is not invented per deal

05 IP and open-source licensing

The question Whether you own what you are selling | **Evidence** Licence inventory, signed assignments, provenance

- [ ] A licence inventory exists, generated from the lockfile rather than from memory
- [ ] Nothing copyleft reaches your shipped or hosted product without a decision behind it
- [ ] Every contributor, employee and contractor has a signed assignment
- [ ] The company account owns the repositories, not a person
- [ ] Vendored and pasted-in code has a recorded origin
- [ ] Every change has a named reviewer on record

06 Data and AI exposure

The question What leaves your trust boundary, and what breaks if a vendor changes | **Evidence** Data map, subprocessor list, a fallback that has been tested

- [ ] You have a map of what customer data leaves your systems, to which vendor, under which agreement
- [ ] The privacy policy permits what the product does, in the words it uses
- [ ] You can name the features that stop working if a model provider changes pricing or terms
- [ ] Swapping a model is a configuration change with a test behind it, not a rewrite
- [ ] Somebody has tried to break your user-facing AI features on purpose
- [ ] If output reaches the EU, the transparency duties that landed in August 2026 are met

07 Roadmap credibility

The question Whether the plan the money funds is buildable by this team | **Evidence** A roadmap whose foundations are visible in the code

- [ ] The next two quarters of roadmap have visible foundations in the current codebase
- [ ] You can show what you cut, and why, on the last three things you shipped
- [ ] Estimates from six months ago can be compared against what happened
- [ ] The hiring plan matches the engineering plan, in both directions
- [ ] Someone owns the roadmap who is not also the person selling it
- [ ] Technical debt has a written list, with the interest rate on each item

The reasoning behind each area, twelve questions an assessor asks first with the shape of a good answer, and what happens when a finding lands are in the full post: **quantumcraft.uk/blog/technical-due-diligence-checklist**. The same checks as a spreadsheet, free, with no form: **quantumcraft.uk/downloads/technical-due-diligence-checklist.csv**.

To see where you stand by area first, take the free self-check: **quantumcraft.uk/due-diligence-self-check**.

If you would rather have an outsider run it, Quantum Craft, a software and AI development agency in London, offers technical due diligence for startups. It ends in a written report that ranks each finding by what it would cost to put right: **quantumcraft.uk/ai-consultancy#technical-due-diligence**.