# Foundations Review: Company X (fictional)

Prepared for: Founders of Company X · Cristian Calabro · Engineering & AI Advisor · ccalabro.dev · Oct 26, 2026

**Sample report. Company X, its people, numbers and findings are fictional, included only to show the format of a Foundations Review.**

## 1. Executive summary

**Company X:** scheduling and invoicing software for small field-service businesses (HVAC, plumbing, electrical). Seed stage, 12 people, about 420 paying customers and USD 1.4M in annual recurring revenue. Engineering is the co-founder CTO, two in-house engineers and a three-person outside agency that builds the mobile app.

**Question:** Can Company X's engineering foundations carry the next 12–18 months: about three times the customers, two more engineers and the planned AI features?

**Verdict: Workable foundations, with gaps to address in the next 90 days (Yellow).** The product is sound and the team ships every week. The risks sit around the code: who can get into what, whether data can be recovered, and how much depends on one person and one agency. None of it needs a rewrite. Most of it is ownership and process.

**Top 3 issues:**

1. **Backups have never been restored, and they share one AWS account with everything else (C1).** 2–3 days. Do it in the first two weeks.
2. **Former contractors still have admin access, and MFA isn't enforced (C2).** 1–2 days. Start this week.
3. **The agency owns the mobile app's code and store accounts (H1).** Bring them in-house within 30 days, while the relationship is good.

**Total effort (estimate):** 32.5–54 engineer-days, about **$26,000–$43,200** at an assumed USD 800/day blended rate. Spread over 90 days, that's roughly a fifth to a third of the in-house team's time. Cloud clean-up (M4) saves about $2,300 a month.

**What's in good shape:** weekly releases with a clear product owner; TypeScript across the API and web app, with strict mode in the API; managed Postgres with daily snapshots and point-in-time recovery switched on; card data handled entirely by Stripe; most infrastructure in Terraform; a strong API test suite around billing; short design notes written before big changes; a shared channel where support flags bugs to engineering the same day.

## 2. Scope and method

Two-week review, Oct 12–23, 2026, covering the technology, how the team works, and AI use. Interviews with both founders, the CTO, the two in-house engineers, the agency lead and the head of customer success. Read access to the four main repositories, the AWS console, CI (GitHub Actions) and both issue trackers. Review of the last 90 days of incidents, pull requests, and cloud and AI bills. Observed one sprint planning session and one code review. Read-only: no changes were made to any system.

| Area | What we looked at | Rating |
|---|---|---|
| Architecture & cloud | AWS hosting (containers on ECS Fargate, RDS Postgres, S3), environments, monthly cost, scaling limits | Yellow |
| Delivery | CI pipelines, release steps, staging vs production, rollback, migrations | Yellow |
| Code quality & testing | Four repositories: API (Node/TypeScript), web (React), mobile (React Native), infrastructure (Terraform); tests, dependencies | Yellow |
| Security & access | AWS and GitHub access, MFA, secrets, offboarding, agency and vendor access | Red |
| Data & reliability | Backups and restore tests, monitoring, alerting, incidents, on-call | Red |
| Team & process | Ownership, code review, planning, onboarding, documentation, key-person risk, agency handoffs | Yellow |
| AI use | AI coding tools and guardrails, the AI feature in the product, data and vendor risk, costs | Yellow |

## 3. Findings, ranked

Severity: Critical act now · High within 30–60 days · Medium within 90 days · Low housekeeping. Effort in engineer-days at USD 800/day.

### Critical

**Critical C1. Production backups have never been restored, and everything lives in one AWS account** · Data & reliability

**What we found:** RDS automated backups (7-day retention) and point-in-time recovery are on, but nobody has ever restored from them and there is no runbook. Backups, production, staging and the CI deploy user all sit in the same AWS account. The S3 bucket with customer job photos and invoices (about 380 GB) has no versioning.

**Business impact:** If the database is corrupted, or deleted with an admin key, recovery time is unknown, and the backups could be deleted along with it. Customers' schedules and invoices are the product.

**Fix:** Run a timed restore to a scratch instance and write a one-page runbook. Copy daily snapshots to a separate backup account with its own access. Turn on S3 versioning. Repeat the restore test every quarter.

**Effort:** 2–3 days · $1,600–$2,400 | **When:** Now (first 2 weeks)

**Critical C2. Former contractors still have admin access, and MFA isn't enforced** · Security & access

**What we found:** Two of the five people who left in the last 12 months (both from the agency) are still in the GitHub organization. One still has an AWS user with full admin rights and an active access key, last used in August, the week they left. MFA is off for 4 of 9 AWS users and isn't required in GitHub or Google Workspace. The AWS root login is shared through a password manager. There is no offboarding checklist.

**Business impact:** A lost laptop or reused password belonging to someone who no longer works with you could reach production and customer data, and nobody would notice.

**Fix:** Remove the two accounts and the key this week. Require MFA on AWS, GitHub and Google Workspace. Put the root login behind hardware keys held by the CEO and CTO, and move day-to-day AWS access to single sign-on. Write a ten-line offboarding checklist and use it for every leaver, agency staff included.

**Effort:** 1–2 days · $800–$1,600 | **When:** Now (this week)

### High

**High H1. The agency owns the mobile app's code and store accounts** · Team & process

**What we found:** The React Native repository lives in the agency's GitHub organization; Company X has read access only. The Apple App Store and Google Play developer accounts are registered to the agency, and the Android signing key is on an agency laptop. The contract doesn't say who owns the code (the agency lead says Company X does).

**Business impact:** About 60% of daily active users are field crews on mobile. If the relationship ends or the agency has a bad month, you can't ship an app update or a security fix.

**Fix:** Transfer the repository to Company X's GitHub, transfer both store accounts (Apple and Google support this), keep signing keys in Company X's secret store, and add ownership and handover terms to the contract.

**Effort:** 2–3 days · $1,600–$2,400 | **When:** 30 days

**High** **H2. API releases run by hand from the CTO's laptop, with no tested rollback** · Delivery

**What we found:** CI builds and tests the API, but deploys (about two a week) run from a script on the CTO's machine. Database migrations run in the same step and can't be reversed. Staging is updated by hand and was three weeks behind production during the review. The last release that had to be undone (Sep 18) took 2 h 40 min to recover.

**Business impact:** Releases stop when the CTO is away, and a bad release means hours of disruption in the middle of your customers' working day.

**Fix:** Deploy from CI on merge to main, with a manual approval for production. Deploy to staging automatically first. Keep the previous version ready to redeploy in one step, and write migrations so the old version still works (add first, remove later).

**Effort:** 5–8 days · $4,000–$6,400 | **When:** Days 31–60

**High** **H3. Customer job notes go to an AI provider with no data review, spend limit or separate keys** · AI use

**What we found:** The "AI job summary" feature (launched in July) sends technicians' job notes to an LLM API. The notes often include customer names, addresses and phone numbers. One API key is shared by development, staging and production and is in the API repository's history. There's no spend limit: AI spend went from about $300 in July to $1,900 in September. Nobody has reviewed the provider's data terms, and the privacy policy doesn't mention the feature.

**Business impact:** When a customer asks where their data goes, there's no clear answer. A leaked key or a retry loop could run up a large bill before anyone sees it.

**Fix:** Rotate the key; use one key per environment with spend limits and alerts. Mask phone numbers and addresses before sending. Confirm the provider's retention and no-training terms and update the privacy policy. Track AI cost per customer per month.

**Effort:** 3–5 days · $2,400–$4,000 | **When:** Now (key), rest in 30 days

**High** **H4. No alerting: customers find outages first** · Data & reliability

**What we found:** Logs go to CloudWatch and errors to Sentry, but no alert reaches a person. Both incidents in the last 90 days were reported by customers: SMS appointment reminders failed silently for 9 hours on Aug 27 after a provider credential expired, and the web app was down for 47 minutes on Oct 2. There is no on-call rotation and no written incident notes.

**Business impact:** Every hour of silent failure means missed appointments for your customers' customers, and support hears about it before engineering does.

**Fix:** Add uptime checks for the app and API, and alerts on error rate, queue age and SMS failure rate to a shared channel and a phone. Set up a weekly on-call rotation across the three in-house engineers, and write a one-page note after each incident.

**Effort:** 3–5 days · $2,400–$4,000 | **When:** 30 days

## Medium

**Medium** **M1. One person holds deploys, billing logic and infrastructure** · Team & process

**What we found:** The CTO wrote about 70% of the billing code, is the only person with production deploy access, and is the only one who knows the Terraform setup. There is no architecture overview.

**Business impact:** A week of illness or vacation stops releases and slows any billing fix. It also keeps the CTO from spending time on hiring and product.

**Fix:** Pair each in-house engineer on a deploy and a billing change. Write a two-page architecture overview and a deploy runbook. Give a second person production access, with MFA.

**Effort:** 4–6 days · $3,200–$4,800 | **When:** Days 31–60

**Medium** **M2. Code review is optional, and the agency approves its own changes** · Team & process

**What we found:** 38% of pull requests merged into the API and web repositories in the last 90 days had no review. Branch protection is off. Mobile changes are reviewed only inside the agency.

**Business impact:** Fewer people understand each change, and risky changes to billing or permissions can ship unseen.

**Fix:** Turn on branch protection: one approving review and passing CI. Billing, sign-in and data-deletion code also needs an in-house reviewer, and every agency pull request gets one too. Add automated reviews: an AI review bot on every pull request (such as CodeRabbit or GitHub Copilot code review), and a review agent developers run locally before opening one. They catch routine issues early; a person still approves.

**Effort:** 1–2 days · $800–$1,600 | **When:** 30 days

**Medium** **M3. Tests are strong in the API and thin everywhere else** · Code quality & testing

**What we found:** API: about 1,100 tests, 71% line coverage, 84% in billing. Web: 46 tests, mostly snapshots, and CI doesn't fail when they fail. Mobile: no automated tests; a manual checklist before each store release takes about a day. The API runs on Node 18, out of support since April 2025, and the four repositories have 47 open dependency alerts, 6 rated high.

**Business impact:** Web and mobile regressions reach customers, and every mobile release costs a day of manual checking.

**Fix:** Make web tests block merges. Add end-to-end tests for the five key flows (book, dispatch, complete, invoice, pay) and a small mobile smoke test. Upgrade the API to Node 22 LTS and clear the high alerts.

**Effort:** 6–10 days · $4,800–$8,000 | **When:** Days 61–90

**Medium** **M4. Cloud spend is about a third higher than it needs to be** · Architecture & cloud

**What we found:** AWS costs about $6,800 a month. Two old environments (demo and qa-old) still run at about $900 a month. Staging runs around the clock at production size (about $1,100). The production database averages 12% CPU on a large instance. About $2,300 a month (34%) can go without affecting customers.

**Business impact:** About $27,600 a year, plus three half-maintained environments that make deploys confusing.

**Fix:** Delete demo and qa-old after a final snapshot. Shrink staging and stop it at night. Right-size the database after two weeks of measurements. Tag resources by environment and set a monthly budget alert.

**Effort:** 2–4 days · $1,600–$3,200 | **When:** 30 days

**Medium** **M5. AI coding tools are used with no shared rules** · AI use

**What we found:** The team uses four different AI coding assistants; two agency developers use personal accounts. There's no guidance on what can be pasted in (customer data, secrets) or on reviewing AI-written code. Tool spend is about $240 a month, paid three different ways.

**Business impact:** Customer data or secrets can end up in tools you don't control, and AI-written code can skip the review other code gets.

**Fix:** Approve one or two tools on company accounts with training turned off. Write a one-page AI use policy: no customer data or secrets, same human and automated reviews (M2), tests required. Add an instructions file for coding agents to each repository.

**Effort:** 1–2 days · $800–$1,600 | **When:** 30 days

### Low

**Low** **L1. A new engineer needs 9 days to ship a first change** · Team & process

**What we found:** The last in-house hire took nine working days to run the stack locally and ship a first change. READMEs describe 2024 setup steps, and the seed-data script is broken.

**Business impact:** Each new hire, including the two planned for next year, loses about two weeks.

**Fix:** One-command local setup with seed data, an up-to-date README in each repository, and a first-week checklist.

**Effort:** 2–3 days · $1,600–$2,400 | **When:** Days 31–60

**Low** **L2. Work is planned in two trackers** · Team & process

**What we found:** In-house work is in Linear, agency work in the agency's Jira, and priorities live in a CEO spreadsheet. 40% of tickets in the last sprint had no acceptance criteria.

**Business impact:** Nobody sees the whole picture in one place, and work gets redone when the goal is unclear.

**Fix:** One tracker for everyone, agency included, a short ticket template with acceptance criteria, and the roadmap kept in the tracker.

**Effort:** 0.5–1 days · $400–$800 | **When:** 30 days

## 4. Effort and cost summary

| # | Severity | Finding | Eng-days | USD | When |
|---|---|---|---|---|---|
| C1 | Critical | Production backups have never been restored, and everything lives in one AWS account | 2–3 | $1,600–$2,400 | Now (first 2 weeks) |
| C2 | Critical | Former contractors still have admin access, and MFA isn't enforced | 1–2 | $800–$1,600 | Now (this week) |
| H1 | High | The agency owns the mobile app's code and store accounts | 2–3 | $1,600–$2,400 | 30 days |
| H2 | High | API releases run by hand from the CTO's laptop, with no tested rollback | 5–8 | $4,000–$6,400 | Days 31–60 |
| H3 | High | Customer job notes go to an AI provider with no data review, spend limit or separate keys | 3–5 | $2,400–$4,000 | Now (key), rest in 30 days |
| H4 | High | No alerting: customers find outages first | 3–5 | $2,400–$4,000 | 30 days |
| M1 | Medium | One person holds deploys, billing logic and infrastructure | 4–6 | $3,200–$4,800 | Days 31–60 |
| M2 | Medium | Code review is optional, and the agency approves its own changes | 1–2 | $800–$1,600 | 30 days |
| M3 | Medium | Tests are strong in the API and thin everywhere else | 6–10 | $4,800–$8,000 | Days 61–90 |
| M4 | Medium | Cloud spend is about a third higher than it needs to be | 2–4 | $1,600–$3,200 | 30 days |
| M5 | Medium | AI coding tools are used with no shared rules | 1–2 | $800–$1,600 | 30 days |
| L1 | Low | A new engineer needs 9 days to ship a first change | 2–3 | $1,600–$2,400 | Days 31–60 |
| L2 | Low | Work is planned in two trackers | 0.5–1 | $400–$800 | 30 days |
| **Total** | | | **32.5–54** | **$26,000–$43,200** | |

Savings not included: about $2,300/month from cloud clean-up (M4), and lower AI spend once limits are in place (H3).

## 5. Decisions

**D1. Stay on managed Postgres (RDS).** The database averages 12% CPU and the data is relational: jobs, crews, customers, invoices. The real risk is recovery, not the engine. Revisit at about ten times today's data.

**D2. Keep the API as one service.** With three engineers, splitting it would add deploys and on-call load without solving a current problem. Today's scaling limits are the database size and the SMS queue, and both can grow in place.

**D3. Two environments: production and staging.** Remove demo and qa-old. Sales demos run on a seeded demo account in staging. Saves about $900 a month and removes confusion about where things are deployed.

**D4. Company X owns every repository, account and key.** Agencies and contractors work inside Company X's GitHub, AWS and store accounts, never the other way round.

**D5. Every change gets one review and passing CI.** Billing, sign-in and data deletion also need an in-house reviewer. An AI review bot checks every pull request, and developers run a review agent before opening one. Small enough to keep, strong enough to spread knowledge.

**D6. Keep the AI job summary, with limits.** Customers use it, so it stays, with masking, separate keys and a spend cap. No new AI features until cost per customer is tracked for a month.

## 6. 30/60/90-day plan

| When | Actions | Owner |
|---|---|---|
| **Weeks 1–2** | Remove former contractors' access, enforce MFA, lock down root (C2). Timed restore test, backup account, S3 versioning (C1). Rotate the AI key (H3). | CTO |
| **Days 1–30** | Move the mobile repo and store accounts to Company X (H1). Alerts, uptime checks and on-call rotation (H4). Review rule, branch protection and automated reviews (M2). Cloud clean-up and budget alert (M4). AI keys, masking and policy (H3, M5). One tracker (L2). | CTO, backend engineer, full-stack engineer |
| **Days 31–60** | Deploys from CI with one-step rollback (H2). Pairing on deploys and billing, architecture overview, runbooks (M1). One-command local setup (L1). | Backend engineer, CTO |
| **Days 61–90** | Web tests block merges, end-to-end tests for key flows, mobile smoke tests, Node 22 upgrade (M3). Second restore test. Check progress against this plan. | Full-stack engineer, agency lead |

## 7. Handover

**What the team receives:**

- This report and the written decisions (D1–D6), with the reasoning behind each.
- The 30/60/90-day plan as tickets in your tracker, each with an owner and acceptance criteria.
- Draft runbooks: database restore, deploy and rollback, offboarding checklist, incident note template.
- A one-page AI use policy draft and an instructions file for coding agents.
- A recorded walkthrough with the founders and the team, then two weeks of async questions.

| Area | Owner after handover | First milestone |
|---|---|---|
| Security & access | CTO | Leavers removed, MFA everywhere (week 1) |
| Backups & restore | Backend engineer | First timed restore (week 2) |
| Delivery pipeline | Backend engineer | Deploys from CI with rollback (day 60) |
| Monitoring & on-call | CTO runs the rotation; all three in-house engineers on it | Alerts live (day 30) |
| Mobile app & agency | CTO owns; agency lead delivers | Repo and store accounts moved (day 30) |
| AI feature & AI use policy | Full-stack engineer | Keys split, masking live (day 30) |
| Cloud cost | CEO reviews monthly; CTO acts | Budget alert set (day 30) |
| Plan progress | CEO | Monthly check against this plan |

Estimates assume a USD 800/day blended rate and the current team doing the work.

If you need help putting it in place, we can work on that together.