Turbo Client Systems, Inc.

Capabilities statement

What we build, how we build it, and what we can be tasked with

Figures as of 2026-11-05

Company overview

Turbo Client Systems (TCS) builds zero-trust, zero-knowledge, Azure-native tools for commercial companies, federal contractors, and federal agencies. The tools sign people in with Microsoft Entra ID and keep customer data in the customer's own Microsoft 365 tenant. Four people run 31 active repositories across four GitHub organizations split by circle of trust (the tools we sell, the shared services, the marketing sites, and the games), one Entra tenant, and one continuous-integration standard.

What we hold about a customer is billing: who pays us for which tools, and nothing beyond what Stripe, our payment processor, holds. The tools run no analytics. We hold what a person chooses to send us (a bug report, a message or booking through a site, or written consent to be named in a case study) and nothing else about their use of a tool.

AI coding agents write nearly all of our code, under rules written by people and checks enforced by machines. A change that passes every required check merges and deploys itself.

TCS is pre-revenue and pre-contract. The known-limits section below states what is not in place yet.

Core competencies

Each capability below is backed by something already built and running, and readable in one of our repositories.

Core competencies, what we can be tasked with, and the evidence

Competency	What we can be tasked with	Evidence already built
Full-stack web application delivery	Next.js 16, React 19, and TypeScript 7 applications with Tailwind 4, shared component libraries, translation, and consent gating	3 public sites and 4 web tools on our own domains; 45 shared packages in one platform repository
Microsoft identity and Graph engineering	Entra ID sign-in in the browser, delegated Microsoft Graph access, a short-lived elevated sign-in for admin-only actions, SharePoint site and list provisioning, and endpoint tables for the commercial, GCC High, and DoD clouds	The sign-in package every web tool uses; the Portal, which provisions a SharePoint site in the customer's tenant
Customer-owned data architecture	Applications that keep customer data in the customer's own Microsoft 365 tenant, with access decided by the customer's Entra groups	Governance and the Portal
GRC and compliance content engineering	A linked graph of statutes, standards, mandates, roles, workflows, exams, and policy templates	Our governance registry, with exams for CMMC Level 2, DFARS CUI protection, NIST CSF 2.0, FISMA, ISO/IEC 27001, HIPAA, HITECH, SOX 404, CCPA, and CFAA
Federal data tooling	Daily collection of public federal opportunity and award data	The Turbo Bid System, which reads SAM.gov Opportunities, Grants.gov, and USAspending each day
Native Windows diagnostics	Tauri 2 and Rust desktop applications that read Windows event logs and system state, with an in-app updater	The Turbo Telemetry System desktop app, version 3.0.0 in development
Delivery governance and CI/CD	Reusable GitHub Actions workflows, a seven-check required gate, merge queues, credential scanning, private package publishing with Changesets, and Dependabot	Every site and tool repository calls the same shared workflow

Platform architecture

One platform repository publishes 45 versioned packages and the shared CI workflow, and every site and tool consumes both. The packages are ES modules with type declarations, published privately to GitHub Packages with Changesets, and consumed as ordinary dependencies, so a fix ships once and every application picks it up through Dependabot.

Identity. Sign-in runs in the browser with Microsoft's MSAL library. By default the token is kept in that tab's session storage; a person can choose to stay signed in across tabs. A rare admin-only action, such as provisioning a SharePoint site, uses a second sign-in held only in memory, with a warning banner on screen while it is live, and discards it afterward. Microsoft endpoints for the commercial, GCC High, and DoD clouds are a typed table in one package, and an unknown cloud fails the build.

Configuration storage in the customer's tenant. The Portal provisions one SharePoint site per customer tenant, the workspace map, holding these lists: Tenant, Products, ProductSettings, Groups, GroupRoles, GroupRoleHistory, ProductInstalls, ManagedTenants, and UserPreferences. Each product serves a JSON manifest at a well-known path declaring the lists, fields, roles, and settings it needs, and reaches its own site through the Sites.Selected Graph permission. A product decides what a person may do from their Entra group membership.

Application shell. Each Next.js application applies a shared configuration that sets Content Security Policy, HSTS with preload, X-Frame-Options, nosniff, a strict referrer policy, and a Permissions-Policy, and mounts Vercel BotID for public endpoints. The consent package gates storage, statistics, and marketing behind a choice the visitor can refuse, a lint rule forbids vendor endpoints written outside the package that owns the vendor, and an opt-in rule forbids tracking that bypasses consent. The UI package is themed through 20 semantic colour roles with light, dark, and high-contrast settings. Our sites send mail through Microsoft Graph with no stored credential, and the payments package refuses Stripe live mode unless it is switched on explicitly.

Hosting. Vercel, one project per repository, deployed only from GitHub Actions; Vercel's own Git integration is off.

Engineering methodology

Every site and tool repository ships through one reusable GitHub Actions workflow, one required-check set, and one merge queue. The workflow is defined once in the platform repository and reaches the other repositories through a tag that moves only after a canary repository passes on the new version. The rules every contributor follows, human or agent, are files in each repository, loaded at the start of every session.

The delivery pipeline. Main requires seven checks to succeed before anything lands there: Generate Artifacts, Security Scan (Gitleaks credential scanning plus actionlint on the workflows), Dependency Rules (import-graph boundaries and cycles), Lint & Format (oxlint and Prettier), Type Check (TypeScript 7, with `any` banned), Unit Testing (Vitest), and Build (a real Next.js build). A pull request into preview needs the first four of them (Security Scan, Dependency Rules, Lint & Format, Type Check), and the merge queue runs them against the exact code that would land. When they are green, the machine merges the change into preview. A person merges the release pull request into main, and a versioned package release.

The path from a ticket to production:
1. Labeled issue 2. Branch and pull request 3. Merge queue runs the fast checks
4. Squash into preview, deployed to a preview host 5. Daily release candidate runs the full suite
6. A person merges the release PR into main; main deploys to production

A change squashes into the preview branch and deploys to a preview host that is not indexed. At 10:00 New York time a release candidate is cut from preview and runs the full suite. Production changes only when a person merges the standing release pull request into main, as a merge commit. Main accepts pull requests only from preview, promotion branches and Dependabot, and it deploys from GitHub Actions in the same run, with the same install step, as the checks.

How we decide what to build and how

- **One stack.** The standard is TypeScript 7, Next.js, React, Tailwind, oxlint, Prettier, Vitest, and pnpm.
- **Duplication over the wrong abstraction.** Code moves into a shared package when it reaches its second or third real use, and the platform repository keeps a ledger of what has moved.
- **Change forward, never roll back.** A bad deploy gets a forward fix through the same checks, including a revert of the merge in a new pull request. No one merges past a failing, pending, or skipped check.
- **Every change links a labeled issue.** Issues carry the same label keys in every repository (size, type, needs, priority, status, env). The needs label says what an issue waits on: an agent, a decision, a look, hands at a keyboard, or money. There is no organization-wide board; the issues are the record.

Dependency hygiene. Dependabot runs in every site and tool repository. Consumer repositories merge green updates automatically; the platform repository batches them on one branch and merges them on purpose, because a bad update there reaches every application.

Agent-assisted engineering under human-authored rules. Most of our code is written by AI coding agents working in isolated git worktrees, claiming work through GitHub issues, and landing it through the same checks as anyone else. The first rules every agent reads are: never fabricate, show your work, and mark what is unverified.

Security, identity, and secrets posture

Workload identity. Our sites send mail with no stored credential: each production deployment exchanges a Vercel OIDC token for a Microsoft Graph token as its own Entra app registration, and Mail.Send is granted only through Exchange scoping to that site's own mailbox. GitHub Actions uses a token minted for each run.

Keys that remain. Some keys still exist. The reviewed key list places eight in GitHub's organization secret store for CI: two for the deploy and the CI app, and six test-account credentials shared only with the repositories that run browser tests. Applications read four at runtime from sensitive Vercel variables. The list holds names only, never values, and the stores themselves were not re-read for this page. Keys are rotated by hand on a calendar until rotation is automated.

Customer data. Governance keeps answers and evidence in a SharePoint site inside the customer's tenant, reached with the signed-in person's own delegated token. Lint rules forbid server actions and any route that reads a request body.

Pipeline and application controls. Gitleaks scans every change for credentials and actionlint checks the workflows themselves. `any` is banned. Every web application ships the same security headers from one shared configuration, and public endpoints carry bot verification.

Products and past performance

We have no contract past performance yet: TCS is pre-contract. Everything below is built, deployed, and run by us. A prime should read this section as demonstrated capability, not as references.

Products, where they run, their status, and what each demonstrates

Product	Where	Status	What it demonstrates
Turbo Governance System	governance.turboclientsystems.com	Upcoming	GRC application that runs as the signed-in person; a crosswalk graph across frameworks; structured exams; storage in the customer's SharePoint; access by Entra group
Turbo Portal System	portal.turboclientsystems.com	Closed Alpha	Customer tenant onboarding; provisions the workspace-map SharePoint site; products register by manifest
Turbo Drift System	drift.turboclientsystems.com	Alpha	Tenant overview through Microsoft Graph as the signed-in admin; a daily snapshot run shipped as a Logic App template that runs in the customer's own Azure subscription
Turbo Bid System	bid.turboclientsystems.com	Upcoming	A daily feed of public federal opportunity and award data from SAM.gov, Grants.gov, and USAspending
Turbo Telemetry System (desktop)	windeets.com	Version 3.0.0 in development	Tauri 2 and Rust; reads Windows event logs and system state on the PC; in-app updater

Azure migration readiness

Already in Microsoft. Identity (Entra ID), authorization (Entra groups), customer data and our own configuration model (SharePoint in the customer's tenant), and site mail (Exchange Online). The Microsoft endpoint package already carries typed tables for the commercial, GCC High, and DoD clouds. We have not deployed to GCC High or DoD, and would prove it in a test tenant before relying on it.

Outside Microsoft today. Hosting is on Vercel (with Vercel BotID, Vercel Web Analytics, which measures a visit only after the visitor consents, and Vercel Blob storage for the Turbo Bid System's feed). Source, CI, and packages are on GitHub, and CI runs on self-hosted runners on our own machines. Moving to Azure means replacing these.

Known limits

We are pre-revenue and pre-contract. These are the limits a prime should price rather than discover.

- Hosting is Vercel's commercial service, not Azure.
- **Not everything is in the trail.** Changes to Azure, GitHub settings and Vercel are not in the git ledger. Every code change leaves a ticket, a pull request and its checks; a change to a live setting does not, and a person makes it.
- **A person promotes every release.** Production changes only when a person merges the release pull request into main. That is a control, because no code reaches customers unreviewed by a decision, and a limit, because a fix waits for that click.
- Our Content Security Policy allows inline script: `script-src` is `'self' 'unsafe-inline'` plus the Vercel Analytics host, so a script injected as inline text would run. A hash cannot cover the per-build chunk list or each page's payload, and a nonce needs every page rendered on each request, which rules out static pages and CDN caching.
- Keys remain (eight in the organization secret store for CI, four that applications read at runtime), and they are rotated by hand.
- Each repository pulls the shared rules with one command, run by hand; nothing pulls them on a schedule.
- CI runs on self-hosted runners on our own machines.
- Unit tests (Vitest) are a required check. The browser accessibility check (Playwright with axe) runs as a separate E2E check in each of our six site repositories, and it is not one of the seven required checks.
- GitHub's paid secret-scanning add-on is not purchased; Gitleaks in CI stands in.
- Governance's audit trail is not append-only yet: a person can still edit or delete their own entries.

- The company is four people.

Differentiators

- **Customer data stays in the customer's tenant.** Governance keeps answers and evidence in the customer's own Microsoft 365 tenant, reached with the signed-in person's delegated token, and lint rules forbid the server code paths that would let it receive them.
- **One platform, one standard.** Identity, consent, security headers, forms, mail, theming, translation, testing presets, and lint rules are versioned packages that every application consumes. A fix ships once and reaches every application through Dependabot.
- **Compliance content is our product domain.** We maintain a linked graph of statutes, standards, mandates, roles, workflows, and exams, including CMMC Level 2, DFARS CUI protection, NIST CSF 2.0, FISMA, ISO/IEC 27001, HIPAA, HITECH, SOX 404, and CCPA.
- **Agent-assisted engineering that a prime can audit.** AI coding agents write nearly all of our code under human-authored rules, through the same checks as anyone else. Every rule, decision, and trade-off is in a versioned file or an issue thread.

How we work with a prime. Everything we build is TypeScript on Next.js with Microsoft Entra sign-in, so a prime's own engineers can maintain it.

Company data

Company registration and contact

Legal name	Turbo Client Systems, Inc.
Location	Merritt Island, FL
UEI	CPYJP28MFBF8
CAGE code	9XZ43
Socioeconomic status	Service-Disabled Veteran-Owned Small Business (SDVOSB); HUBZone certified
Contract vehicle	GSA Schedule, IT services (confirmed 2026-08-04; contract number not yet recorded)
Security clearances	Active clearances held by team members (confirmed 2026-08-04)
Point of contact	John Roller, President
Email	contracts@turboclientsystems.com
Phone	321-376-7160
Web	turboclientsystems.com

NAICS codes

Code	Description
238210	Electrical Contractors and Other Wiring Installation Contractors
541330	Engineering Services
541420	Industrial Design Services
541512	Computer Systems Design Services
541519	Other Computer Related Services
541990	All Other Professional, Scientific, and Technical Services