

Code Due-Diligence Summary

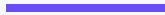
Anonymous Company Platform Assessment

Prepared by The Code Registry for WhiteGloveCustomer

April 2026 | Full report: v1.0.3



Contents



01	Executive Summary
02	Estate Overview
03	Security & Risk Assessment
04	Technology & Operational Risk
05	People & Process
06	Risk Register & Recommendations



01

Executive Summary

Overall assessment, key takeaways, and methodology



THE CODE
— REGISTRY —

Diligence supports a 12-18 month migration and sunset strategy



Engagement Goal

Transition to WhiteGloveCustomer systems

Strategic Alignment

The diligence confirms the estate is supportable for the target 12-18 month window. This enables WhiteGloveCustomer to:

- Operate safely during the bounded migration period.
- Focus resources on customer transition rather than long-term modernization.
- Execute a disciplined sunset of the legacy backend.

Key Decision Support

Takeaways provided by this assessment mitigate immediate risks to the sunset timeline:

- Identifies backend security as the primary containment priority.
- Highlights knowledge concentration as the main continuity risk.
- Validates that no material blockers exist for the proposed acquisition.

The legacy backend is the primary risk-bearing system; the estate is supportable for migration but not low-risk without active containment

The Anonymous Company platform can be operated safely during a bounded migration period, but carries a meaningful cluster of risks that should be treated as active migration-period concerns. **See full report Section 1 (pp. 3–7).**

1

Primary Risk Concentration

The legacy *AnonymousCompany Online Web App* is the primary operational and risk-bearing system — not the Flutter client repositories.

2

Material Risk Cluster

Legacy backend security weaknesses, secret/configuration hygiene, migration-operability fragility, technology supportability, and knowledge concentration.

3

Migration Supportability

The estate appears supportable for a bounded migration period, but **not low-risk by default** without active containment and monitoring.

4

Dependency & OSS Profile

Most dependency and open-source issues appear **secondary** to the legacy backend's security, continuity, and operational risk profile. The main concern is external service reliance, not package vulnerabilities.

5

Remaining Open Questions

Anonymous Company has provided final production context on controls, infrastructure, service ownership, and operational support. **No material open questions remain.** The diligence view is now fully grounded.

Source: Anonymous Company – Code due-diligence report [v1.0.3], Section 1 (pp. 3–7)

Five repositories were reviewed using evidence-led, risk-oriented methodology focused on migration-operability

Repository	Type	Source Control	Apparent Role	Diligence Importance
AnonymousCompany Online Web App	Legacy web platform	TFVC	Core legacy application	High
anonymous-company-customer	Flutter app	Git	Customer mobile app	Medium
anonymous-company-studio	Flutter app	Git	Studio mobile app	Medium
anonymous-company-shared	Shared library	Git	Shared domain/resources	Medium-Low
app-components	Shared components	Git	Reusable UI components	Low-Medium

Methodology

The review is **evidence-led and risk-oriented**, focused on whether the platform can be operated safely during migration — not whether it is a long-term modernization candidate.

The legacy web app was reviewed from a TFVC ZIP export (Git history unavailable). The four Git-based repositories support full commit and contributor analysis.

Source: Anonymous Company – Code due-diligence report [v1.0.3], Section 2 (pp. 8–10) and Section 3 (pp. 11–13)

Evidence Sources

- The Code Registry platform analysis outputs
- Repository-level composition and language analysis
- Dependency, license, and security scan outputs
- Targeted Code IQ exploratory queries
- Anonymous Company production context responses

02

Estate Overview

Architecture, repository structure, and runtime dependencies

The estate is a hybrid architecture with a legacy .NET backend supporting modern Flutter client applications

Legacy Backend Platform

AnonymousCompany Online Web App

- ASP.NET Web Forms / VB.NET / .NET Framework 4.8
- Monolithic architecture with tightly integrated business logic
- Server-rendered UI, runtime DLL dependencies, reporting components
- Multiple externally consumed API capabilities
- Principal operational system for core business workflows

Modern Client Layer

anonymous-company-customer &

anonymous-company-studio

- Flutter / Dart mobile and web client applications
- Native Android and iOS entry points
- Consume remote services via API-driven integration
- No direct compile-time coupling to legacy internals
- Environment-configured backend API access

Shared Libraries: *anonymous-company-shared* (domain models, API client, common services) and *app-components* (reusable UI components, form controls, styling). These are not independently deployable but are architecturally important internal dependency nodes.

Migration Perspective: The frontend clients are separated from backend implementation details by configuration and service boundaries — a positive characteristic. The principal migration-period operational dependency risk is concentrated in the legacy backend, which contains the densest concentration of core business workflows: **payments, messaging, scheduling, reporting, webhook processing, and background operational services.**

The legacy backend carries the densest concentration of business-critical integrations and external service dependencies

Business-Critical Third-Party Services

- **Stripe** — Primary payment processor
- **EVOSnap / Global Payments** — Active, in sunset planning
- **Payscout / Paywire** — Payment processing
- **Twilio** — Messaging and communications
- **Mandrill** — Email delivery
- **Azure Blob Storage** — File and asset storage
- **reCAPTCHA** — Bot protection
- **Zapier** — Webhook and automation integrations
- **Crystal Reports** — Reporting and data export

Key Dependency Findings

- All service accounts and credentials are held in **company-owned** @anonymousecompanysoftware.com accounts — not tied to personal or contractor accounts.
- **Mercury** is legacy and not in use; **Firebase** is present in code but not yet active in production.
- **No private package feeds** or proprietary binary dependencies are required to build or run the platform.
- The *flutter_stripe* fork in anonymous-company-customer was created for a documented upstream Android performance issue. The fix has now been merged upstream (Feb 2026), making replacement feasible with rigorous

Migration implication: External service dependencies affect continuity, account ownership, operational support, and migration sequencing. EVOSnap/Global Payments and Crystal Reports are the most notable operational pain points.

Runtime Package Vulnerability Profile

The estate does not show a large volume of runtime-vulnerable packages. The main validated runtime concern is the **dio** dependency in anonymous-company-studio (actively used as the production HTTP client). Many other flagged dependency issues were determined to be build-time only or otherwise not materially reachable in production runtime paths.

03

Security & Risk Assessment

Security posture, secrets handling, and repository hygiene

The most material security risks are concentrated in the legacy backend, with SQL injection, legacy password hashing, and exposed secrets

LEGACY BACKEND FINDINGS

- **SQL injection risk** — Raw SQL query concatenation without parameterization identified in first-party production code
- **Legacy password hashing** — Salted SHA256 (not aligned with modern best practice; bcrypt/PBKDF2 recommended)
- **Hardcoded encryption keys** — Symmetric keys embedded in source
- **Exposed secrets in code** — Secret material present in repository contents
- **Limited anti-CSRF coverage** — Only partial confirmed protection across authenticated flows
- **Broad privileged attack surface** — Admin portal on same public domain as customer-facing app, protected at application layer only (3 authorized users)

FLUTTER LAYER FINDINGS

The Flutter repositories present a **lower but still relevant** set of client-side security hygiene concerns:

- *anonymous-company-customer*: Token-based auth with local storage, weak XOR-based secret protection, tokens in local storage without strong encryption, no explicit MFA or password-strength enforcement in client code
- *anonymous-company-shared*: **Exposed Slack webhook URL** — confirmed secret-management failure with material risk of unauthorized messaging or operational disruption
- *anonymous-company-studio*: Token-based auth, local secret storage in Hive without strong encryption, runtime-reachable vulnerable **dio** dependency (production HTTP client)

COMPENSATING CONTROLS

- API and long-running services hosted on **Azure App Service**
- Legacy apps hosted on **Azure Windows VM**
- Direct VM and database access **IP-restricted** to two named personnel
- **MFA enforced** for Azure portal accounts; no personal accounts used
- Access logging via **Azure Activity Log**
- **Gaps**: No WAF, no EDR, no SIEM in place. Azure portal itself is not IP-restricted.
- Anonymous Company reports **no prior security incidents** and has not commissioned an independent penetration test.

Secret management is mixed across the estate, with plaintext credentials in legacy web.config representing the most significant hygiene concern

LEGACY WEB APP — WEB.CONFIG

- **Cleartext connection strings with passwords** in web.config — present in source control
- Connection strings and configuration values remain stored in plaintext within web.config on the VM-hosted environment
- Exposure is bounded by IP-restricted VM access and MFA-protected source-control access
- **Anonymous Company priority:** Rotation and externalization of web.config secrets identified as a high-value near-term action

CROSS-REPOSITORY HYGIENE CONCERNS

- **Duplicated API keys** — ANONY_API_KEY appears across multiple platform-specific environment files (production and staging), increasing blast radius
- **Exposed Slack webhook URL** in anonymous-company-shared — confirmed secret-management failure
- **Local token storage** in Hive without strong encryption across Flutter apps
- **Deployment tokens** in build-related configuration artifacts (Fastlane Google deployment material, App Store Connect keys, Azure Variable Groups)
- **Positive patterns:** Pipeline-based injection, runtime configuration, and limited use of public-only identifiers in cleaner repositories (e.g., app-components)

AZURE APP SERVICE COMPONENTS

- Secrets managed through **Azure application settings**
- **Encrypted at rest** by Azure
- This represents a directionally stronger pattern than the legacy web.config approach
- The mixed secret-management model confirms that credential rotation and externalization are appropriate near-term containment actions

04

Technology & Operational Risk

Technology lifecycle, dependencies, technical debt, and migration operability

Technology lifecycle risk is moderate at the infrastructure level but concentrated in the legacy application architecture

PRODUCTION INFRASTRUCTURE STACK

Legacy Web Platform

- .NET Framework 4.8
- IIS 10.0.20348.1
- Azure SQL Database
- Windows Server 2022 Datacenter Azure Edition v21H2

Mobile Application Layer

- Flutter current stable channel
- Android minSdkVersion 21, targetSdkVersion 35
- iOS deployment target 14.0

INFRASTRUCTURE STATUS

- Anonymous Company confirms **no components are currently out of support** or in extended support only
- SQL Server to Azure SQL Database migration completed successfully in **December 2024**
- No failed or too-risky upgrade attempts reported
- No specialist third-party support contracts beyond standard Azure / Microsoft coverage
- **Crystal Reports remains active** in production and is required for all reporting functionality — this constrains deployment options for the legacy web platform

Key diligence distinction: The main lifecycle concern is not immediate end-of-support status, but **application-level supportability**. ASP.NET Web Forms, VB.NET, Crystal Reports, and legacy operational patterns are harder to patch, staff, and troubleshoot under migration time pressure.

Build-tooling clarification: Node, Python 3.14, and Ruby 3.4.7 are confirmed to affect only test, build, or distribution tooling — not production runtime. This reduces uncertainty around non-Microsoft build-chain risk.

Dependency risk is driven by external service reliance rather than a large volume of runtime-vulnerable packages

RUNTIME PACKAGE VULNERABILITIES

- **anonymous-company-studio:** Runtime-reachable vulnerable **dio** dependency — actively used as the production HTTP client for API interactions
- **anonymous-company-customer:** Ruby gems in Gemfile.lock are **build-time only** — not reachable in production runtime
- **anonymous-company-studio (previously flagged):** Axios SSRF and Handlebars RCE concerns are confined to **test-only code** — deprioritized from live operational risk
- **app-components:** ATS and Subresource Integrity warnings affect only **example/demo content**

EXTERNAL SERVICE RELIANCE

The greater concern is reliance on **business-critical external services** in the legacy backend:

- **Payment processors:** Stripe, EVOSnap/Global Payments (in sunset planning), Paycom/Paywire
- **Messaging:** Twilio, Mandrill
- **Storage:** Azure Blob Storage
- **Reporting:** Crystal Reports (operationally required)
- **Integrations:** Zapier, reCAPTCHA

All service accounts are **company-owned**. No private package feeds or proprietary binaries required.

flutter_stripe fork update: The fork in anonymous-company-customer was created for a documented upstream Android performance issue. The relevant pull request was merged into the official package in **February 2026**. Replacement is now feasible, subject to rigorous payment regression testing.

Internal dependency structure: anonymous-company-customer and anonymous-company-studio both depend on anonymous-company-shared for shared application logic and API-facing behavior, and on app-components for UI consistency. This shared-library coupling means migration changes affecting models, API contracts, or service abstractions could have **cross-application impact**.

Migration-operability fragility is concentrated in payment, messaging, scheduling, webhook, and background-processing workflows

TECHNICAL DEBT CONCENTRATION

AnonymousCompany Online Web App — highest-risk maintainability hotspot:

- Mixed C#, VB.NET, ASP.NET Web Forms stack
- Average complexity score materially above other repositories
- Densest concentration of business-critical functionality

anonymous-company-customer — second-tier concern:

- Extremely concentrated ownership (single dominant contributor)
- Creates meaningful bus-factor and support survivability risk
- Overlap with customer-facing workflows increases migration relevance

MIGRATION-OPERABILITY RISK PATTERNS

Recurring risk patterns across the estate:

- **Synchronous external calls** with long timeouts (120-second waits in shared API layer)
- **Retry and cache-invalidation loops** that can amplify backend load
- **Polling-based workflows** in billing and transaction handling
- **Tightly coupled billing models** increasing migration sensitivity
- **Background workers / queue-like processing** for operational continuity
- **Calendar event fetching, pagination, and attendance retry invalidation** in studio app

Five most brittle workflow categories under migration stress: (1) Payment and post-payment processing, (2) Messaging and scheduled email delivery, (3) Scheduling and calendar publication, (4) Webhook and event-driven processing, (5) Reporting and export-heavy administrative activity.

Anonymous Company confirms these workflows are **stable under production load** and subject to Azure monitoring and alerting. The key open question is whether monitoring depth, documentation quality, and failure-threshold visibility are sufficient for a migration period in which coexistence and operational pressure may increase.

Source: Anonymous Company - Code due-diligence report [v1.0.3], Section 9 (pp. 34–37) and Section 10 (pp. 38–40)



05

People & Process

Contributor concentration, knowledge continuity, and delivery practices



THE CODE
— REGISTRY —

Knowledge is heavily concentrated across the Git-based repositories, creating meaningful bus-factor risk during migration

CONTRIBUTOR CONCENTRATION FINDINGS

Git-based repositories show **very high authorship concentration**:

- *anonymous-company-customer*: One dominant contributor accounts for **nearly all recorded activity**
- *anonymous-company-shared*: Extremely concentrated ownership
- *app-components*: Very high ownership concentration
- *anonymous-company-studio*: Slightly broader contributor base, but still meaningfully concentrated around a single dominant contributor

For the legacy TFVC web platform, repository-history visibility is limited — continuity risk is assessed through Anonymous Company's confirmed support model and knowledge-retention context.

CONFIRMED KNOWLEDGE HOLDERS

Anonymous Company confirms the following knowledge distribution:

- **Dave Rudger** — Full knowledge of core legacy platform (payments, scheduling, messaging, reporting, customer data). Can independently support.
- **Matt Hansen** — Full knowledge of the platform. Can independently support.
- **Mark** — Full knowledge of App Service applications; can be brought up to speed on Flutter maintenance.
- **Guy Reardon** — Full knowledge of the Flutter applications.
- **Jett Stucky** — Working knowledge of admin portal and customer-support workflows.
- **No former employees or contractors required** to keep the platform running.

TSA mitigation: Contributor concentration is partly mitigated by the TSA structure retaining Dave Rudger for **12 months post-close**, together with WhiteGloveCustomer's stated intention to retain Matt Hansen post-close. However, background job configuration, payment-processor integration details, Crystal Reports structure, and WebForms deployment process remain **only lightly documented**.

Delivery practices: Release flow for AnonymousCompany Online Web App is feature branch → dev → staging → production, with an accelerated staging-hotfix path. Releases are managed by Dave and Mark. The process is functional but **not formally documented**. No critical fixes are known to exist outside the current production line.

Source: Anonymous Company - Code due-diligence report [v1.0.3], Section 11 (pp. 41–44)

Open source license exposure is low-to-moderate, with the flutter_stripe fork and a small GPL footprint as the main attention areas

The dominant license types across the estate are standard permissive licenses (MIT, Apache 2.0, BSD). This does not appear to be a headline transaction blocker, but follow-up items remain. **See full report Section 12 (pp. 45–47).**

flutter_stripe Fork

anonymous-company-customer

Manually maintained fork under MIT license. Created for a specific upstream Android performance issue. Fix now merged upstream (Feb 2026). Remaining concern: safe replacement with rigorous payment regression testing.

GPL Footprint

anonymous-company-studio

Small number of GPL-licensed components detected. Not yet a confirmed major copyleft problem. Warrants confirmation of how components are used and whether they are part of anything distributed externally.

Vendored Packages

AnonymousCompanyOnline

Vendored packages included directly in repository for self-contained builds. Typical .NET / NuGet dependency environment with permissive OSS and Microsoft proprietary licensing. Need clarity on provenance and what is shipped.

CLEAN REPOSITORIES

anonymous-company-shared and *app-components* appear comparatively clean from an OSS and license perspective. *anonymous-company-shared* uses standard Dart / Flutter packages under permissive licenses with no vendored or forked components detected. *app-components* shows a dependency profile dominated by standard permissive licenses with no significant vendored or forked component concerns in production.

Overall severity assessment: Low-to-moderate. The main diligence significance lies in normal attribution and packaging review rather than any identified high-severity OSS compliance defect. Final legal conclusions depend on how software is packaged, deployed, and redistributed in practice — not just the code inventory itself.

Source: Anonymous Company - Code due-diligence report [v1.0.3], Section 12 (pp. 45–47)

06

Risk Register & Recommendations

Prioritized risks, containment actions, and next steps

Day-one containment should prioritize legacy backend security, secret rotation, and monitoring of integration-heavy workflows

Priority Tier	Recommended Actions	Anonymous Company Alignment
Day-One Containment	• Confirm which hardcoded values are live production secrets and rotate any live values • Validate production reachability of identified SQL injection vectors • Confirm anti-CSRF coverage across authenticated flows • Begin credential rotation and removal of web.config values from source control history	Identified as high-priority near-term actions. Password-hashing migration to bcrypt/PBKDF2 viewed as planned improvement rather than day-one emergency.
Migration-Period Operational Priorities	• Increase monitoring and alerting on background workers, webhooks, and scheduled jobs • Monitor payment and webhook processing for queue buildup or retry thrashing • Plan for knowledge continuity around Crystal Reports and payment-processor integrations • Sequence migration activity around the workflows most likely to become brittle under stress	6–12 month stated priorities include increased monitoring and documentation of release process, hotfix procedure, and background job dependencies.
Lower-Priority Monitoring / Cleanup	• Password-hashing migration to bcrypt or PBKDF2 as a planned improvement • flutter_stripe fork replacement with upstream package and rigorous payment regression testing • GPL compliance confirmation in anonymous-company-studio • OSS attribution and packaging review for AnonymousCompanyOnline vendored packages	Known internally as planned improvements. Minimize changes to shared libraries during migration.

Key alignment: Anonymous Company's stated 6–12 month priorities closely match these recommendations, including credential rotation, monitoring improvements, and documentation of critical operational processes.

Source: Anonymous Company - Code due-diligence report [v1.0.3], Section 13 (pp. 48–51)

The Code Registry

Enterprise Code Intelligence

thecoderegistry.com

For the full 60-page analysis, see

Anonymous Company – Code due-diligence report [v1.0.3]



THE CODE
— REGISTRY —