Certify Intel (Excel-Only) — Codex Handoff Plan

Version: v3.1 handoff • Date: 2026-01-14 • Repository: Project_Tzu (Private)

1) Problem Statement (What we are building)

Build a comprehensive, automated market and competitor intelligence solution implemented entirely within a single Excel workbook for Certify Health, with a premium Certify-branded UI and dashboards, autonomous competitor discovery, competitor product/service identification, data scraping, pricingg, compeiti and evidence ingestion from allowlisted sources, strict fail-closed extraction that never infers missing facts, versioned evidence-backed claims with full traceability, middle-market segmentation enforced across all surfaces, advanced analytics, and an operational workflow that can be run repeatedly and reproducibly by non-technical users.

2) Solution Summary (What we built and how it satisfies the problem)

The current solution is an Excel-only intelligence engine with an evidence-led data model (entities, sources allowlist, evidence, claim candidates, claim versions, events, alerts, review tasks, and run logs). A macro-driven pipeline orchestrates optional competitor discovery, evidence fetching, deterministic extraction, fail-closed promotion, change detection, and alert evaluation. Dashboards and technical views dynamically surface key KPIs, coverage, and UNKNOWNs. Packaging scripts convert the workbook container into a macro-enabled deployment and import VBA modules reproducibly.

3) Repository and Deliverables (What belongs in GitHub vs outside)

3.1 Repo contents (source only)

The GitHub repo is source-only: /vba (all .bas modules), /scripts (installer and packaging), /docs (architecture/data model/pipeline/runbook/security), and /assets (logo and branding references). Excel binaries (.xlsx/.xlsm), zipped packages, exports, API keys, and any client data are excluded by design.

3.2 Workspace conventions (local-only)

Local workspace path used for deployment and testing: C:\Users\conno\Downloads\Certify_Health_Intelv1. Store deliverables under a local deliverables subfolder; do not commit them.

4) Completed Work (Current state as of v3.1)

1. Excel workbook container implementing an evidence-led schema and operational tabs that support repeatable runs (Input, Metrics, Dashboard, Technical Dashboard, Review Queue, Events, Alerts, Run Log, Claim Versions, Evidence Snapshots, etc.).

2. Fail-closed logic enforced: missing/ambiguous values remain UNKNOWN and are routed to Review_Queue; promotions require valid evidence_id and non-UNKNOWN values; allowlist checks gate evidence creation.

3. Self_Tests sheet updated to validate key invariants (no promoted candidate without evidence, no promoted UNKNOWNs, allowlist_ok enforcement, no blank claim statuses).

4. Added Feature_Taxonomy and Feature_Matrix scaffolding to support feature overlap analytics and heatmap-ready competitor comparisons.

5. Workbook stability hardening: replaced spill-based data validation patterns with stable named ranges backed by a Lists sheet to prevent Excel repair/removal of validations and formulas.

6. White background enforced across all tabs and logo placement standardized across sheets (branding consistency).

7. Packaging bundle created: installer scripts that convert .xlsx → .xlsm and import VBA modules into the workbook reproducibly; dashboard buttons are created by macro during installation.

8. Created core repo documentation drafts (architecture, data model, pipeline, security, runbook) and repo governance guidance (private repo, no license, no binaries, no secrets).

5) Remaining Work (To fully complete the project as intended)

1. Competitor discovery provider hardening: implement and standardize on an actively supported search provider (e.g., SerpAPI/Brave/Foundry) with explicit licensing compliance; update macros and docs accordingly.

2. Multi-source ingestion: add adapters for tiered sources (company site, press releases, SEC filings where relevant, reputable directories, review sites) with source-tier classification; RSS/social treated as signals only until corroborated.

3. Deterministic extraction expansion: extend rule-based extraction coverage for pricing model, contract terms, ACV/ARR proxies, feature set, vertical focus, geography, size, funding/valuation, integrations, and implementation model; all fields must be evidence-span linked.

4. Entity resolution workflow: canonicalization, fuzzy matching, merge/split workflow with admin audit trail; prevent duplicate competitors and maintain stable entity IDs.

5. Middle-market segmentation engine: encode explicit segment rules (revenue/practice size/region) with an auditable rule table; enforce segment filters on every dashboard/analysis/export.

6. Analytics layer completion: implement explainable calculations for growth proxies, feature overlap scoring, pricing vs value, clustering, regression, and market share proxies; store results with traceable inputs and evidence linkage.

7. Dashboard and technical dashboard refinement: finalize UX navigation, slicers/filters, KPI tiles, coverage gauges, trend charts, and drill-through to evidence and claim history.

8. Events and change detection: expand evidence diffing beyond hash-only by capturing stable text extracts/snippets; generate classified events (pricing change, feature change, messaging change) with review tasks.

9. Alerting integration: optional Outlook/Teams notification via VBA (no secrets stored); configurable alert rules by severity and segment scope.

10. QA and regression testing: broaden Self_Tests into a full invariant suite, add a repeatable sample dataset harness, and define Definition of Done checks before packaging.

11. Release engineering: versioned packaging (zip), checksum, changelog updates, and a consistent install/upgrade pathway that preserves local data tables where required.

6) Codex Agent Plan (How to complete the remainder efficiently)

6.1 Agent roles (recommended)

1. Agent A (Tech Lead / Architect): owns requirements traceability, segmentation rules, source-tier policy, and acceptance criteria; reviews all PRs.

2. Agent B (VBA Pipeline Engineer): owns macro pipeline orchestration, installer compatibility, allowlist enforcement, logging, and error handling.

3. Agent C (Extraction & Evidence Engineer): owns deterministic parsers, evidence span capture, candidate scoring, and UNKNOWN routing logic.

4. Agent D (Analytics & Excel Modeling Engineer): owns Metrics/Technical Dashboard formulas, scoring models, explainability tables, and visualization integrity.

5. Agent E (QA / Release Engineer): owns Self_Tests expansion, sample harness, packaging reproducibility, and documentation/runbook accuracy.

6.2 Codex working rules (must-follow)

1. Never add Excel binaries, exports, zips, executables, scraped data, API keys, or secrets to the repo.

2. All new claims must have: evidence_id, source tier, evidence spans/snippet, confidence, status; missing data must be UNKNOWN + Review_Queue task.

3. All pipeline steps must be idempotent where possible and append-only where history matters (Claim_Versions).

4. Every change must include updated docs and Self_Tests when it affects invariants or workflow.

5. Always return diffs and keep PRs small (single responsibility).

6.3 Suggested execution workflow in Codex

1. Open repo (Project_Tzu) in Codex Web and confirm folder structure (/vba, /scripts, /docs).

2. Start with invariant-hardening PR: sheet/table existence checks, defensive error handling, and run-log accuracy.

3. Implement discovery provider standardization and update docs; add Self_Tests that validate provider configuration states.

4. Expand deterministic extraction library by field group; add unit tests via sample harness; route all uncertainties to Review_Queue.

5. Complete analytics and dashboards only after extraction coverage is sufficient; add explainability tables for each score.

6. Finalize release packaging scripts and runbook; cut a version tag and produce a client deliverable outside the repo.

7) Top 5 Next Tasks (Most important, in logical order for Codex)

1. Pipeline hardening and invariants: add sheet/table existence checks, strict input validation, and comprehensive run logging; expand Self_Tests to cover all fail-closed rules end-to-end.

2. Discovery provider switch and compliance: implement one supported provider (SerpAPI or Brave) with explicit allowlist/tier rules; update Config schema, VBA, and docs; add self-test for missing/invalid keys that creates a review task without breaking other runs.

3. Deterministic extraction expansion (Tier-1 facts): implement parsers for pricing model/ACV/contract signals, core feature set, vertical focus, region, integration claims; require evidence spans/snippets and UNKNOWN routing for any ambiguity.

4. Segmentation engine (Middle Market enforcement): build an auditable rules table and a deterministic segment classifier; ensure all dashboards, analytics, and review queues are filtered by segment_id with no bypass.

5. Analytics + dashboard completion: implement explainable scoring models (threat/opportunity/coverage), feature overlap heatmaps, change-over-time views from Evidence_Snapshots/Events, and drill-through from dashboard tiles to evidence and claim versions.

8) Ten Additional High-Value Features (Aligned with original goals)

1. Evidence viewer pane: a dedicated sheet that displays the evidence snippet, URL, and highlighted span for the selected claim candidate or claim version.

2. Merge/split wizard for entities: UI-driven canonicalization, alias management, and merge audit trail to prevent duplicate competitor rows.

3. Source policy manager UI: a controlled interface to manage allowlisted sources, tiers, and crawl limits with an auditable change log.

4. Signal-to-fact promotion workflow: RSS/PR/social entries land as signals and can only be promoted to facts when corroborated by tier-1 evidence; enforced by macros and Self_Tests.

5. Competitive positioning map: 2x2 and multi-axis maps (pricing vs value, feature breadth vs depth, implementation speed vs complexity) with explicit explainability tables.

6. Market share proxy module: configurable proxy selection (web traffic, review volume, job postings, customer logos) with per-proxy confidence and evidence linkage.

7. Change classification: convert raw evidence diffs into typed events (pricing change, messaging change, feature launch, vertical expansion) with severity and suggested action.

8. Outlook notification integration: optional VBA-based alert emails for high-severity events, generated from the Alerts table and logged in Run_Log (no secrets stored).

9. Lockdown/role modes: a read-only client mode that locks structure and macros; an admin mode for schema edits and allowlist changes.

10. Export-ready audit pack: generate an internal audit sheet bundle (claims, evidence, versions, exceptions) to support review meetings and client sign-off.

9) Definition of Done (Completion criteria)

1. Workbook runs end-to-end with one button: discovery (optional), evidence ingest, extraction, promotion/versioning, events, alerts, dashboards refresh.

2. All claims visible in dashboards are evidence-backed with version history and drill-through to evidence.

3. Middle market segmentation rules are explicit, auditable, and enforced everywhere; no unsegmented analytics.

4. Self_Tests pass on a clean run and fail appropriately when invariants are violated.

5. Packaging is reproducible: install script produces a working .xlsm with all macros/buttons; install instructions are correct.

6. Repo remains source-only with no binaries, secrets, or client data committed.

Repository File Boundaries (Mandatory for Codex Agents)
This repository is source-only. Codex agents must assume that the following files and directories **exist in the repository and are the only files they are permitted to read or modify**:. gitignore, README.md, /docs/*, /vba/*.bas, /scripts/*, and /assets/*. Codex agents must also assume that **Excel workbooks (xlsx, .xlsm), packaged artifacts (.zip, .exe), installer outputs, exports, scraped data, runtime logs, and any files containing API keys, secrets, or client data DO NOT exist in the repository and must never be created, modified, referenced, or committed**. All Excel binaries and client deliverables are stored outside of version control in a local workspace only and are intentionally excluded from GitHub. Any attempt to introduce or rely on non-repo files violates project constraints and must be rejected.