# Tactiq

AI football match-analysis platform | Designed, built and operated solo by Murat Can Ümit
https://www.tactiq.club | https://www.mcvibeapps.com

## OVERVIEW

Pre-match and live tactical analysis, outcome probabilities and supporting statistics for individual fixtures, aimed at fans who want reasoning rather than a single number. One Flutter codebase across iOS, iPadOS, Android and macOS. 32 languages. Subscriptions priced across 175 countries using World Bank purchasing-power data. Over 20,000 users. A free tier plus three paid tiers, reconciled server-side. Most engineering effort sits in the prediction and language layers rather than the interface, because the analysis is the product.

## PREDICTION ENGINE

A layered pipeline rather than a single model. Different questions are answered by different mechanisms, kept separate so each can be measured and changed in isolation.

- **Team strength rating.** Elo-style rating combined with a competition-seed composite for tournaments. Elo updates incrementally from results and needs no external ranking feed; the seed composite exists because a new tournament field has no shared match history to rate from.
- **Expected goals with caps and floors.** Poisson model over expected-goals values produces scoreline and market probabilities. A strength-difference term overrides model-produced expected goals when they collapse into a narrow band, because the language model compresses toward a draw-like range. Hard caps and floors bound the output so one extreme input cannot produce an implausible distribution.
- **Ensemble structure.** The rating engine and the language model each produce an implied 1X2 view and are compared rather than blended. Each fixture is tagged confident, uncertain or disagree, because the disagreement is itself signal: agreement raises trust, disagreement flags caution rather than a false-precise merge.
- **Form weighting.** Directional change over the last seven matches against the season baseline, fed in as a prior. Directional rather than absolute because a mid-table team improving and a strong team declining are different bets that an average would hide.
- **Head-to-head tiebreaker.** Qualification and elimination scenarios use head-to-head and permutation logic rather than points, because in group and knockout stages table position does not determine who advances, and treating it as if it did produced misleading must-win language.
- **Match context.** Adjusts for knockout weighting where draws resolve through extra time and penalties, venue and home advantage, and standings relevance. The same two teams behave differently in a neutral-venue knockout than in a league fixture.

## MACHINE LEARNING LAYER

Built in-house in pure Python. Scikit-learn, gradient boosting and deep-learning runtimes were deliberately not adopted: serverless functions need small memory footprints and fast cold starts, the models must stay interpretable so a wrong result traces to a cause, and at this data scale a transparent model correctable per league beats an opaque trained one that is hard to debug when a single league drifts.

- **Online rating updates.** Team ratings update from every graded result with no offline retraining step, so the strength model moves with recent reality rather than a static seed.
- **Calibration feedback loop.** Every past prediction is graded against its actual outcome, scored with a Brier metric and a reliability breakdown by confidence band, and fed back into both probability shaping and automatic rollback thresholds.
- **Style clustering.** Hand-implemented k-means groups teams by playing style, so similar teams inform each other’s priors where direct match history is thin.

The principle is that the learning stays legible: every number a user sees traces to a rating, a calibrated probability or a cluster, each of which can be inspected and corrected. Credibility depends on being right in a way that can be explained, and fixable when it is wrong.

## LLM LAYER AND GUARDRAILS

Amazon Bedrock with two model tiers: a cheaper model via cross-region inference profiles for the free tier, a stronger model on the global profile for paid tiers. Output tokens dominate cost, so the stronger model is reserved for users whose spend justifies it.

- **Server-authoritative grounding.** Team, player and tournament names and statistics are never read from model output. They are injected from verified data after generation and the model is instructed to treat them as fixed. The model structurally biases toward famous clubs and invents plausible but wrong entities, so identity is treated as data rather than something the model may decide.
- **Confidence floor.** Generated probabilities are bounded so the model cannot express certainty the data does not support. An overconfident wrong call damages trust more than a hedged one.
- **Structured output contracts.** A fixed JSON schema, validated before use. A result failing validation or post-checks is never written to cache, because a bad cached result is served to every later reader of that key, turning one error into many.
- **Language guard.** Generated prose is classified against the requested language using script dominance for non-Latin scripts and stopword and diacritic scoring for Latin ones. On mismatch it regenerates once at temperature zero with a correction directive, and if it still fails it is not cached. For a 32-language product, answering in the wrong language is a correctness failure rather than a cosmetic one.
- **Fallback and decline.** A status gate blocks analysis where a pre-match prediction would be meaningless. Guard failures fail open on the user path but fail closed on security and quota: a guard error should never silently deny a paying user, and an auth error must never grant free access.

## CALIBRATION AND MONITORING

A weekly monitor, scheduled Sunday at 22:00 UTC, joins graded fixtures back to their predictions and reports hit rate for the rating engine against the language model, broken down per league, per tier and per confidence band, alongside calibration error, latency and null rate. Results are written to dedicated calibration record and summary tables rather than logs, so a historical read is always available. The language model runs as a live baseline rather than an absolute target, because beating a competent baseline is the honest test of whether the extra machinery earns its place.

Drift detection uses calibration error and per-league accuracy. Automatic rollback is wired to hit-rate floors so a regression reverts without human intervention. A module or league is demoted only after a per-league read on a real sample, not on one alarm.

A worked example: a calibration alarm fired on a thin off-season sample of around eighty graded events. I chose not to refit, because the high-confidence bucket held only ten events. Two weeks later, on roughly four hundred events, calibration measured as good and the restraint was confirmed correct. The rule that came out of it is to verify sample size before acting on an alarm, because a small sample produces alarms that look like model failure and are not.

## DATA LAYER

1,228 competitions, with a featured subset of roughly two hundred receiving deeper enrichment: injuries, expected-goals inputs, shot profiles. Full-depth enrichment everywhere would exceed provider quotas for leagues almost nobody queries. Reconciliation runs at both write and read time. Group-stage standings, for instance, are de-duplicated by team because the provider doubles rows once results arrive, and the read path cleans stale shapes so a fix takes effect without waiting for a re-sync. Freshness is matched to volatility: fixtures twice daily, standings and injuries on multi-hour cycles, live scores every minute. Missing fields resolve to safe defaults or an empty state the interface renders honestly, and provider schema changes are absorbed by defensive normalisation, because a silently changed field upstream should degrade to a blank and never to a fabricated value.

## PLATFORM ARCHITECTURE

One hundred and seventy-six Lambda functions across three regions in an active-active layout behind latency-based routing with sixteen Route 53 health checks, so two regions can fail and the third still serves. The distribution is deliberate rather than symmetric: us-east-1 carries eighty functions and runs the batch, calibration and tournament workloads, while eu-central-1 and ap-northeast-1 carry forty-four and fifty-two and serve the user path. Runtime is split roughly evenly between Python 3.12 and Node.js 24. The function count is high by design: prediction, language, calibration and data layers are separate functions so each deploys, tests and rolls back independently, which matters more than minimising surface when correctness sits on the revenue path.

State lives across forty-one DynamoDB tables. Thirty-six replicate to all three regions as Global Tables. Five are held outside global replication deliberately: paid entitlement and refresh tokens sit in a single-region authority with one writer using atomic conditional writes keyed on the last event timestamp, then propagated outward by a reconciler. Multi-region last-writer-wins can silently drop a tier change, and a customer losing paid access is the failure the whole design exists to avoid.

Every request passes three fail-closed checks before any logic runs: app attestation (Play Integrity on Android, App Attest on Apple), an HMAC signature over the request, and a signed token. Infrastructure is Terraform, with CI running syntax, tests and localisation completeness gates on every push.

## MONETISATION AND ENTITLEMENT

Four subscription products plus promotional grants. Usage is metered as a per-tier daily allowance rather than a purchasable balance, decremented by atomic conditional write and only on successful generation, because charging for a failed request is both wrong and a support cost. Entitlement is reconciled server-side and never trusted from the device: a store event reaches a webhook, updates the single-region authority, and propagates to the multi-region user tables, so the client displays state it cannot itself grant. Purchases span the Apple ecosystem, where iOS and iPad share a purchase and macOS is a separate Mac App Store product, and Google Play, reconciled into one cross-platform entitlement. Plan changes follow store semantics, upgrades immediate and downgrades deferred to period end, with refunds and chargebacks on the same event path and a weekly reconcile catching drift. Prices are set per country from purchasing-power data rather than flat currency conversion, because a flat price is unaffordable in some markets and undersells in others.

## COST CONTROL

Two hard budget caps sit on the account with alarms attached: three hundred dollars a month for Bedrock inference and five hundred for everything else. They are not forecasts. Setting the ceiling before scaling forced the two-tier model routing, the caching rules and the decision about which requests justify a stronger model. A generative product without a ceiling discovers its unit economics in a billing email.

## LOCALISATION

32 languages with no machine translation in delivered strings, because a football audience notices wrong terminology immediately and a mistranslated tactical term reads as incompetence. Three surfaces are managed separately: in-app strings with a CI completeness gate so no language ships partial, store listings, and legal texts. The runtime language guard closes the loop, since native static strings do not help if the model answers in English.

## FEATURED WORK: WORLD CUP 2026 HUB

Built as an isolated fleet separate from the main system, so a high-traffic seasonal event could be built, tuned and later removed without touching the core. It added a knockout-weighting engine covering extra time, penalties and advancement probability that never treats a knockout as a possible draw; a fix hiding cross-group standings comparisons in the knockout stage, because comparing teams from different group tables is misleading; and a controlled live-analysis setting where many users repeatedly analysed the same fixtures, producing dense calibration data that fed back into the main engine. The measurable outcome was a per-league accuracy and calibration record from real tournament traffic, and a set of improvements validated there before being considered for the wider league season.