Football Matches Tracker — Architecture v1.0

Date: 2025-08-16
Owner: Vitalii (single admin user)
Hosting: Netlify (publish=public, functions=functions)
DB: Neon Postgres
Frontend: Vanilla HTML/JS/CSS
Functions: Netlify Functions (CommonJS, `exports.handler`)

> Note: This document fixes the agreed architecture. No secrets or private data included.

---

0) Scope & Goals

- Private, login-gated single-user app (only **admin** exists now).
- Robust against free-tier abuse; all endpoints (except `login` and internal scheduler) require auth.
- Data model supports later expansion to regular `user` role without migrations.
- Safe, idempotent, TV-friendly UI (Sony Bravia / Android TV, D-pad).
- Sync mechanism adds/updates matches **without** losing `seen/comments` and **without** duplicates.

---

1) Roles & Access Control

- Roles present in schema: `admin`, `user`. **Currently only** `admin` **is used.**
- All browser endpoints require **session auth**.
- Access matrix (final):
- **Public:** `POST /login` only; `POST /scheduled-update-matches` is **internal** (scheduler).
- **Authenticated (admin):** all other endpoints.
- If `user` ever appears: they may `updateMatch` / `setPreference`, not `setSort` / `update-matches`.

---

2) AuthN (Sessions) & AuthZ (Roles)

Users table keeps `role CHECK IN ('admin','user')` (using only `admin` now).
Sessions: secure cookie with HMAC; server-side session row.

Session Cookie

- Format: `session=<sid>.<sig>` where `sig = HMAC(SESSION_SECRET, sid)`.
- Attributes: `HttpOnly; Secure; SameSite=Lax; Path=/; Max-Age=30d` (rolling: refresh every ≤12h of activity).
- Rotation: new `sid` after login and periodically.

Endpoints

- `POST /login` → create session cookie.
- `POST /logout` → revoke session, clear cookie.
- `GET /me` → `{ authenticated, role, csrf }` if logged in; used at app boot and after login.

CSRF (Double Submit Token)

- Per-session `csrf_token` signed with `CSRF_SECRET`.
- Returned via `GET /me` (or `getPreferences`).
- Required for all state-changing browser requests except `login/logout`.

---

3) Endpoints (Contract-Level)

All require auth + CSRF unless stated.

- `POST /login` (public) → set cookie; returns `{ ok: true }` on success.
- `POST /logout` (auth) → revoke session; CSRF optional.
- `GET /me` (auth required for app; returns `{ authenticated, role, csrf }`).
- `GET /getMatches` (auth): returns match list.
- `GET /getPreferences` (auth): returns `{ sort_col, sort_order, seen_color, csrf }`.
- `POST /updateMatch` (auth, CSRF, rate-limited): toggle `seen`, update `comments` (and admin granular edits; see Manual Overrides).
- `POST /setPreference` (auth, CSRF, rate-limited): update `seen_color`.
- `POST /setSort` (admin, CSRF, rate-limited): update global sort.
- `POST /update-matches` (admin, CSRF, rate-limited, lock+idempotency): manual sync.
- `POST /scheduled-update-matches` (internal scheduler): same sync logic, internal guard, no CSRF.

Responses on rate limit / lock:
- `429 Too Many Requests` + `Retry-After` on per-endpoint limits.
- `204 No Content` (or `429`) when `sync_lock` active; body `{ status: "locked_until", locked_till }`.

---

4) Sorting Model (Query-Level)

- Sorting is **only** in SQL `ORDER BY`, not stored positions.
- Global prefs in `preferences(sort_col, sort_order)`.
- Stable tie-breaker:
  `ORDER BY <sort_col> <dir>, kickoff_at DESC, home_team, away_team, id`.
- New rows naturally fit current rules; existing rows keep relative order unless source fields change.

---

5) Data Model (DB Schema Spec)

5.1 `matches` (primary table)

- **Identity**
- `id` PK
- `kickoff_at` timestamptz (UTC)
- `date_bucket` timestamptz = `date_trunc('minute', kickoff_at AT TIME ZONE 'UTC')`
- `home_team`, `away_team` text (displayed as is)
- `home_team_canon`, `away_team_canon` text (canonicalized via alias map)
- `pair_key` text = `sort([home_team_canon, away_team_canon]).join('|')`
- **UNIQUE** `(date_bucket, pair_key)` (order-independent uniqueness)

- **Source (updatable) fields**

- `tournament` text
- `link` text (url)
- `link_version` int (start 1), `link_last_changed_at` timestamptz
- `rank` int
- (opt) `status` text enum: `scheduled|live|finished|postponed|cancelled|hidden`
- (opt) `home_away_confidence` enum: `high|medium|low`

- **Local (never touched by autosync)**

- `seen` boolean default false
- `comments` text (sanitized)

- **Service/Audit**

- `updated_at` timestamptz default now()
- `updated_by` int nullable (FK users.id; null for autosync)
- `manual_overrides` JSONB (keys of columns frozen from autosync)
- (opt) `updated_cols` text[] (last change info)

- **Indexes**

- `idx_matches_kickoff_at`
- `idx_matches_rank_desc`
- `idx_matches_tournament` (opt)

5.2 `preferences` (1 row)

- `sort_col` (`kickoff_at|rank|tournament|…`), `sort_order` (`asc|desc`)

- `seen_color` text (preset/hex)
- `updated_at`, `updated_by`

5.3 `settings`

- `key` PK text, `value` text/jsonb
  Use: `sync_lock` (`{ locked_till: timestamptz }`), feature flags, etc.

5.4 `users`

- `id` PK, `username` UNIQUE, `password_hash`, `role` CHECK IN (`admin`, `user`), `created_at`, `last_login_at`

5.5 `sessions`

- `sid` PK, `user_id` FK, `issued_at`, `expires_at`, `revoked` boolean

5.6 `rate_limits`

- `key` PK (e.g., `sess:<sidprefix>:<endpoint>` / `ip:<canon>:<endpoint>` / `global:<endpoint>`)
- `count` int, `reset_at` timestamptz, (opt) `last_seen_at`

5.7 `sync_logs`

- `id` PK; `started_at`, `finished_at`, `status` (`ok|skipped|failed`)
- `inserted`, `updated`, `skipped` ints; `source` text (`manual|scheduler|import:<name>`)
- `actor_user_id` nullable FK; `idempotency_key` text; `note` / `error` text

5.8 `match_changes` **(optional but recommended)**

- `id` PK; `match_id` FK; `changed_at` timestamptz
- `source` (`auto|manual`); `changed_by` nullable FK
- `diff` JSONB (e.g., `{ "link": {"old":"…","new":"…"}, "rank": {"old":3,"new":1} }`)

5.9 `match_links`

- `id` PK; `match_id` FK; `link` text; `changed_at` timestamptz; `source` (`auto|manual`)

5.10 `staging_matches`

- Only **source** fields (+ `import_batch_id` UUID, staging metadata).
- TTL cleanup or batch cleanup after merge.

---

6) Canonicalization & Uniqueness

- **Alias map** for team names → `*_canon` columns.

- `pair_key = sort([home_team_canon, away_team_canon]).join('|')`
- Uniqueness key: `(date_bucket, pair_key)` (order-independent; prevents duplicates).
- Display keeps `home_team` / `away_team` as last accepted values.
- Autosync may change `home_team/away_team` **only** when `home_away_confidence='high'` (or manual patch).

---

7) Sync Process (No-Duplicate, No-Loss)

Inputs: normalized list with source fields (`kickoff_at`, `home_team`, `away_team`, `tournament`, `link`, `rank`, optional `status`, `home_away_confidence`).
Steps: 1) **Validate & normalize** → compute `date_bucket`, canonical names, `pair_key`; fill `staging_matches` with `import_batch_id`. 2) **Lock**: read `settings.sync_lock`; if `now < locked_till` → return 204/429 w/o work. Otherwise set `locked_till = now()+Δ` (3–5 min). 3) **Idempotency**: check `sync_logs` by `Idempotency-Key` (UI provided, scheduler slot key, or payload hash). Duplicate → return previous result. 4) **UPSERT** `staging` → `matches` by `(date_bucket, pair_key)`: - Update **source fields only** per rules below; never touch `seen/comments`. - `tournament`, `rank`: always update. - `link`: update; `link_version++`; add row to `match_links`; set `link_last_changed_at`. - `kickoff_at`: update if $|\Delta| \leq 2$h **or** `home_away_confidence='high'`. - `home_team/away_team`: update only with `high` confidence and **not** in `manual_overrides`. - Respect `manual_overrides` JSONB: skip any key present. 5) **Log** to `sync_logs` (`inserted/updated/skipped`, `source`, `actor_user_id`, `idempotency_key`). 6) **Cleanup**: delete `staging` for `import_batch_id` (or TTL clear over time).

Dry-run: optional `?dry_run=1` computes `{ would_insert, would_update, would_skip }` without writes.

---

8) Manual Overrides & Granular Edits

- Admin UI can PATCH source fields per match: `{ link?, tournament?, rank?, kickoff_at?, home_team?, away_team? }`.
- For every edited field, set override flag in `manual_overrides` (JSONB).
- Autosync skips overridden fields until cleared (manual clear or TTL e.g., 30 days).
- Audit to `match_changes` with `source='manual'`, `changed_by`, and column diffs.

---

9) Rate Limits (Fixed Window)

Storage: `rate_limits(key, count, reset_at)` (10-minute windows unless noted).
Keys: `sess:<sidprefix>:<endpoint>`, fallback `ip:<canon>:<endpoint>`, plus `global:<endpoint>`.

- `POST /updateMatch`, `POST /setPreference`: Session 60/10m; IP 60/10m; Global 600/10m.

- `POST /setSort`: Session 20/10m; IP 20/10m; Global 200/10m.
- `POST /update-matches` (manual): Session 1/5m; IP 1/5m; Global 1/2m + `sync_lock` 3–5m + Idempotency.
- `POST /scheduled-update-matches`: Global 1/5m.

On 429: return `Retry-After` seconds; UI disables controls for that duration.

Circuit breaker (soft): if approaching provider quotas, temporarily halve heavy limits; respond with 429 + explanatory message.

---

10) Privacy / GDPR

- **No raw IP/UA stored** by default.
- If required: store `ip_hash`, `ua_hash` using `HMAC(PRIVACY_SALT, value)`.
- **Retention (TTL):**
- `sessions`: delete expired >30 days.
- `sync_logs`: 90 days (or 30 stricter).
- `rate_limits`: remove keys after `reset_at + 24h`.
- `match_changes`: 90 days.
- **Vendors:** Netlify (hosting), Neon (DB). Legal basis: legitimate interests.
- **Privacy page**: explain data, purpose, retention, vendors, and contact for requests.

---

11) Security Headers / CSP

- Keep strict CSP in `public/_headers` (no inline JS; only required origins).
- `Permissions-Policy` deny camera/mic/geolocation; `Referrer-Policy: strict-origin-when-cross-origin`.
- `frame-ancestors 'none'` (or `X-Frame-Options: DENY`).
- `X-Content-Type-Options: nosniff`.
- CORS: `Access-Control-Allow-Origin: APP_ORIGIN` only; `Vary: Origin`.
- Cookies: `HttpOnly; Secure; SameSite=Lax`.

---

12) TV (Sony Bravia / Android TV) UX Guidelines

- D-pad focus order; no hover dependencies.
- Large targets (≥48×48dp), clear focus outlines; fonts ≥18px.
- Natural focus cycle: Header controls → Table rows (Seen → Link → Comment).
- Links: use `<a href target="_blank" rel="noopener">`; ensure CSP allows navigations; avoid `window.open` popups.
- Back button returns focus; disable heavy animations; keep layout margins for overscan.

---

13) Admin UI — Interactions

- **Header**: “Admin”, `Update Matches`, `Sort`, `Color`, `Logout`.
- **Update Matches**: modal with optional **Dry-run**. Handles `200 ok`, `200 duplicate`, `204/429 locked`, `400 bad_source`, `500 failed`. Shows small summary and countdowns.
- **Sort**: immediate save; refresh table on success; rate-limited.
- **Color**: presets; immediate save; re-render on success.
- **Row actions**: Seen toggle (optimistic), Link open, Comment inline edit.
- **Edit…**: granular PATCH of source fields + override toggles; clear override button.

---

14) Observability & Errors

- `sync_logs` capture counts, durations, source, actor, idempotency key.
- `match_changes` keeps diffs for manual changes.
- No secrets in logs. Classify errors (transient vs permanent). Single backoff retry for transient DB failures.

---

15) Configuration (ENV)

- `APP_ORIGIN`
- `DATABASE_URL`
- `CSRF_SECRET`
- `SESSION_SECRET`
- `ADMIN_USERNAME`, `ADMIN_PASSWORD_HASH`
- (opt) `PRIVACY_SALT`
- (opt) `SCHEDULER_SECRET` / header for internal scheduler guard

> Never print or commit any secrets.

---

16) Testing Strategy (Methodology)

Goals: verify contracts, auth/CSRF, rate limits, sorting stability, sync rules, manual overrides, and TV navigation basics.

Layers

1) **Unit tests (functions/utils)**
- CSRF token generation/verification.
- Session cookie parsing/signing.
- Rate-limit key math & reset logic.
- Canonicalization (`pair_key`, alias map).
- UPSERT decision rules (pure functions: what to update/skip per input & overrides).

2) **Integration tests (Netlify Functions local)**
- Run with `netlify dev` (or direct handler import) + **Neon test branch** or local Postgres (Testcontainers).
- Cover login → me → protected POST with CSRF; verify 401/403/429 flows.
- Sync pipeline: staging load → upsert → logs; idempotency duplicate run.
- Manual overrides respected on next autosync.

3) **End-to-End (E2E)**
- **Playwright** headless desktop profile to simulate D-pad (arrow keys) navigation.
- Verify focus order, enter/back actions, modals, toasts, and link activation.
- Smoke on TV browser is manual but E2E ensures regressions are caught.

Environments

- **DB**: use **Neon branching** for `test` (ephemeral branches per CI run), or **Testcontainers Postgres** locally.
- **Secrets**: `.env.test` with test creds; **never** reuse prod values.
- **Data fixtures**: minimal seed (few teams, 3–5 matches) + synthetic sync payloads.

Tooling

- **Unit/Integration**: Vitest (or Jest) + Supertest/undici for HTTP.
- **E2E**: Playwright.
- **Coverage**: thresholds for critical modules (auth/csrf/sync rules).
- **Static checks**: ESLint, Prettier (optional).

CI

- **GitHub Actions**: on PR/push:
  1) Install Node LTS.
  2) `npm ci`.
  3) Spin Neon test branch (API) or Testcontainers.
  4) Run unit+integration.
  5) Run a small headless Playwright suite.
  6) If all green → allow merge; Netlify deploys `main`.

Test Matrix (Essentials)

- Auth: good/bad credentials; expired session; logout.
- CSRF: missing/invalid token → 401/403.
- Rate limit: per endpoint; `Retry-After` honored.
- Sorting: ORDER BY stability (no jumps with equal keys).
- Sync: new rows insert; existing update; `kickoff_at` ±2h; `link` versioning; `home/away` only at `high`; idempotency duplicate.
- Manual override: set & clear; autosync skip respected.
- Privacy: no raw IP/UA persisted.
- Headers: CSP present; cookies have secure attributes.

---

17) Migration Plan (from current DB)

1) Add new columns to `matches`: `kickoff_at`, `date_bucket`, `home_team`, `away_team`, `home_team_canon`, `away_team_canon`, `pair_key`, `link_version`, `link_last_changed_at`, `updated_at`, `updated_by`, `manual_overrides`.
2) Backfill from existing `match` text → split into `home/away`, derive `kickoff_at` from `date`, compute `date_bucket` & `pair_key`; initialize `link_version=1`.
3) Create UNIQUE `(date_bucket, pair_key)`; add indexes.
4) Create/adjust `preferences`, `settings`, `users`, `sessions`, `rate_limits`, `sync_logs`, (`match_changes`, `match_links`, `staging_matches`).
5) Switch functions to new auth/CSRF and new rules; remove legacy `ADMIN_TOKEN`, old `_db.js` after cutover.

---

18) Next Steps (Execution Checklist)

- [ ] Confirm alias map for team canonicalization (initial seed).
- [ ] Define confidence assignment for `home/away` (source rules).
- [ ] Prepare DB migrations per spec above.
- [ ] Implement new endpoints (`login/logout/me`) and update middlewares.
- [ ] Enforce auth on all endpoints; replace Bearer with cookie sessions.
- [ ] Implement sync lock + idempotency + dry-run.
- [ ] Wire rate limits per endpoint.
- [ ] Update UI flows (login screen, header controls, row actions).
- [ ] Add minimal tests (unit→integration→E2E).
- [ ] CI gate before Netlify deploy.

This document is the source of truth for the agreed architecture.
When implementation deviates, update this doc first, then code.