ESPN Eredivisie Fantasy (EEF) Toolkit — PRD v0.3 (Next.js)

ESPN Eredivisie Fantasy (EEF) Toolkit — Product Requirements Document (v0.3 • Next.js-Oriented)
(Living document – updated 19 July 2025)

0 OVERVIEW
Vision: Provide FPL■level (and beyond) analytical & planning capability for EEF; architect it now so it can later open to other users without a rewrite.
Current User: Single hardcore manager (you). Future■ready architecture.
Primary Outcome (v1 Next.js): Desktop Next.js app serving pre-generated datasets + internal normalized API (no direct raw endpoint calls client-side).
Data Source Strategy: Reverse engineer raw → ingest → normalize → internal API JSON → Next.js pages.
Scope (v1): Squad planner (formation), FDR (attack/defence) ticker & rankings, price monitor, enriched player tables, i18n (NL/EN).
Out of Scope: Auth, notifications, projected points, websockets, advanced ML, mobile, monetisation, community features.

1 GOALS & SUCCESS
Functional completeness; Data integrity (<3% missing); Predictor accuracy (≥80% rises/drops); Performance (Lighthouse ≥90, FCP <1.5s, TTI <2.5s); Build reliability; Personal usability (plan ≤10 min).

2 TARGET USER
Single advanced fantasy manager (you). Multi-user later with minimal refactor.

3 FUNCTIONAL REQUIREMENTS
3.1 Data Ingestion & Internal API
- Cron (≥2× daily) fetches raw endpoints (bootstrap-static, fixtures, team, etc.).
- Versioned raw JSON stored under data/raw/DATE/TIME.
- Normalize → internal/*.json (players, fixtures, price_history, fdr_attack, fdr_defence).
- Enrich → players_enriched.json (player + FBref + price deltas + FDR aggregates).
- Manifest (schemaVersion, dataVersion, timestamps, counts, hashes). Build fails if stale or mismatch.

3.2 Squad Planner
Formation: XI on pitch; 1 GK; DEF 3–5; MID 3–5; FWD 1–3; total 11.
Bench: 1 GK + 3 outfield ordered.
Constraints enforced on drag/drop (invalid revert with feedback).
Initialization: Import current squad (team ID) OR blank build.
Scenario Management: Multiple named scenarios (local); save, overwrite (confirm), delete, export/import JSON, autosave on change.
Captain/Vice required exactly one each (warning if invalid).
No projected points (explicit exclusion placeholder).

3.3 FDR & Horizon Rankings
Attack & defence FDR per GW (prior-season xG For / xG Conceded base).
Custom horizon N (3,5,6,8,10, custom) average attack & defence per team.
Ranking tables ascending easiest (tie-break variance or home match count).
Player view: offensive positions show attack FDR aggregate; GK/DEF show defence FDR aggregate.
Files: fdr_attack.json, fdr_defence.json, fdr_matrix.json.
Toggle combined / attack / defence.

3.4 Price Change Monitor
Compute price & ownership deltas (24h & velocity). Heuristic flags rise/drop. Sortable/filterable. Highlight owned players.

3.5 FBref Integration
Nightly scrape (≥3s delay) → stats → mapping → enrichment.

3.6 Localization
i18n en/nl JSON; toggle persisted.

3.7 Ownership & Template Tracker (Backlog)
3.8 League Enhancer (Backlog)
3.9 Accessibility & Theming (Must)
Accessible pitch (labels & keyboard). Color safe.
3.10 Data Quality Dashboard
Raw vs internal counts; unresolved FBref mappings; formation validation issues.

4 NON-FUNCTIONAL
Architecture: Static internal API consumption (no runtime protected endpoint calls).
Performance: Pitch drag frame ≤16ms avg.
Maintainability: Internal API schema version → regenerate TS types.
Accessibility: Keyboard bench <-> pitch moves.

5 ARCHITECTURE
Raw Ingestion → Normalization → Type Gen + Build → Static Next.js → Client Planner (local).

6 DATA PIPELINE
Fetch Raw → Normalize Players → Price History → FDR Calc → FBref Scrape → Enrich → Manifest.

7 DATA MODEL (TS - summary)
PlayerCore, PlayerFBref, PlayerDerived, PlayerEnriched, SquadSlot, SquadScenario, TransferRecord.

8 COMPONENTS
Layout, NavBar, LanguageSwitcher, PlayerTable, FDRTicker, FDRRankings, Planner, PitchFormation, BenchBar, ScenarioList, ScenarioToolbar, PriceChangeTable, DataQualityPanel, FormationWarning.

9 ROUTES
/, /players, /planner, /price-changes, /fdr, /internal/data

10 STATE
Zustand slices: preferences, scenarios, planner. Versioned schema & migration.

11 I18N
en.json, nl.json; build check missing keys.

12 PERFORMANCE
Pre-computed metrics; code splitting; virtualization; hashed JSON assets.

13 TESTING
Unit (formation validator, FDR calc), Component (tables), E2E (scenario CRUD & invalid drag), Data integrity (manifest).

14 STACK
Next.js (App Router), TypeScript, Tailwind, Zustand, GitHub Actions, Vercel, Vitest, Playwright.

15 RISKS
Formation edge cases → tests. Local storage corruption → export/import & schema version backup. API changes → modular parser. Mapping errors → unresolved list & manual overrides.

16 MIGRATION
Import old Streamlit JSON → SquadScenario.

17 NEXT ACTIONS
1 endpoints doc 2 ingestion scripts 3 type gen 4 /players page 5 formation validator 6 pitch + bench 7 scenario CRUD 8 FDR calc & rankings 9 price table 10 i18n toggle 11 CI + deploy.

18 GLOSSARY
Internal API, Formation Validator, Scenario, Manifest, Horizon (GW range).