SportIQ — PRD Change Summary (Updated V1 + Partial Backend)

Audience: Mobile-responsive UI and Backend Development Teams

Executive Summary

SportIQ evolved from a Player-focused, frontend-only prototype (localStorage) into a demo-ready, role-complete application with Coach, Admin, and Government modules, plus a partial backend for Auth, Profile, and Details. The project maintains a legacy-local-first architecture for most features while introducing a Node/Express + Prisma (SQLite) backend for identity and profile, bridged back into localStorage to preserve existing flows. The system now includes tournament creation, government approval, publication workflows, registration management, scheduling, and a new Coach Reports landing page built in Tailwind with the Stitch AI design language.

Key architectural decisions include a cookie-based session, a stable user schema, and a strict policy baseline (e.g., government approval always required before publish; duplicate registrations blocked; lastDate enforced). The Reports landing page is local-source for KPIs with an optional session sync and has placeholders to integrate a future /reports API without UI changes.

From a development standpoint, the largest impacts are:
- A hybrid data model (DB-backed Users; localStorage for all other entities)
- Expanded role guards/policies
- New Coach/Admin/Government feature surfaces
- Clear normalization rules and data-ownership boundaries
- A modular, Tailwind-driven UI layer (mobile-first) consistent with Stitch design tokens

---

Chronological Change Log

ID	Phase	Area	Change	Rationale	Impact
C1	V1 baseline	Frontend	Player features (Profile, Achievements, Tournament finder) with localStorage persistence	Rapid prototyping; no backend dependency	All data in localStorage under sportiqData
C2	V1 baseline	CSS/UX	CSS architecture Step 1: global.css with reusable glassmorphic components	Consistency and reuse	Lower CSS duplication; clearer component tokens
C3	V1 baseline	Onboarding	3-step Details: OTP (123456), strict 10-digit mobile, role/sport, optional avatar	Ensure dashboards and flows behave correctly	Fields persisted on user; redirect to login

C4	Updated V1	Roles/Features	Implemented Coach (Verify, View Players, Schedules), Admin (Create Tournament, Manage Registrations), Government (Verify Tournaments), Player (Find/Register/My Tournaments)	Complete role coverage for demo	New pages, modules, and guards across app
C5	Updated V1	Data Model	Added top-level arrays: schedules, scheduleRequests, tournaments; embedded registeredTournaments on User	Support schedules and tournament lifecycle	Broader schema; cross-entity queries in localStorage
C6	Updated V1	Auth/UX	Login via email or username; show password; tolerant state/district matching	Improve usability	Minor controller updates, better input handling
C7	Policy decisions	Governance	Govt approval always required before publish; officials can browse any region but act only within their region	Clear regulatory workflow	Stricter publication pipeline; UI permissions
C8	Policy decisions	Registration	Disallow duplicate registrations; enforce registration.lastDate	Data integrity and fair process	Input validation and guard logic
C9	Policy decisions	Achievements	Keep “edit Approved” allowed for now; plan disabling later; allow any proof file type (demo)	Minimize friction; expand acceptance	UI preview/handling for non-images required
C10	Updated V1	Formatting	RegisteredTournament.date: “DD MMM YYYY – DD MMM YYYY, HH:mm”; reminder default false	Consistent snapshots; predictable UI	Formatting utilities; default flag
C11	Backend intro	Backend/Infra	Node/Express + Prisma (SQLite), cookie session (sid JWT), endpoints: register/login/logout/me GET/PATCH; static /public; port fallback	Stabilize identity and profile	DB-backed Users; no server endpoints yet for other entities
C12	Backend bridge	Integration	Mirror backend user into localStorage after Auth/Details/Profile; preserve achievements/registeredTournaments	Keep legacy pages working	Hybrid data ownership; local guards remain
C13	Normalization	Data rules	DOB ISO in DB, yyyy-mm-dd in local; height/weight kept as string; mobile must be 10 digits strict	Predictable types; validation	Input parsing; server-side validation
C14	Migration policy	Accounts	Legacy local-only users must re-register; assume fresh backend IDs; no ID remapping	Reduce complexity	No one-time merge/mapping logic required
C15	Coach Reports	UI	New Tailwind-based Reports landing page (Stitch tokens). KPIs computed locally with optional backend session seeding	Provide analytics landing quickly	Modular store/UI/controller; future /reports API ready

---

Detailed Impact Assessment and Rationale

- Functional (Mobile-responsive UI)
- Implemented end-to-end role experiences: Coach (Verify, Players, Schedules), Admin

(Tournament wizard, Registrations), Government (Verification), Player (Find/Register/My Tournaments).
- New Coach Reports landing page with Stitch AI visual language, mobile-first grid (1/2/3 columns), sticky header/filter bar, skeleton loading, deep-link placeholders. This targets core KPIs: Attendance (30D), Approved vs Pending achievements, Active players (7D), Upcoming sessions (7D), Registrations (PENDING/CONFIRMED).
- Strict guard behavior: legacy pages use local guards; Coach-only access enforced where applicable.

- Backend System Architecture
- Introduced Node/Express + Prisma with a single User model and cookie-based sessions (JWT in httpOnly sid). Static frontend served via /public with auto-port fallback for CodeSandbox.
- Backend endpoints are restricted to Auth and Profile (register/login/logout/me GET/PATCH). All other features remain localStorage-driven.
- “Bridge” mirrors backend User into localStorage, ensuring legacy features operate unmodified and requireLogin continues to function.

- Technical Implementation Shifts
- Data model expanded: schedules, scheduleRequests, tournaments at top-level; registeredTournaments embedded snapshots on User.
- Normalization: DOB (ISO in DB; yyyy-mm-dd in local), mobile strict 10-digit validation, numeric fields kept as strings in local.
- Policy tightening: Govt approval becomes mandatory before publish; region-scoped actions for officials; duplicate registrations disallowed; registration deadline enforced.
- Achievements proof type broadened to “any file type” (demo), requiring UI to degrade gracefully for non-image previews.

- Specific Feature Changes
- Admin publication pipeline: Admin creates !’ SUBMITTED !’ Govt approves !’ Admin publishes !’ PUBLISHED (Govt “Pending” = SUBMITTED).
- Manage Registrations: publish/unpublish, PENDING/CONFIRMED/REJECTED workflows, CSV export remains demo.
- Schedules: free-text venues; entrance flag stored (no functional differences in V1).
- Reports: local-first computations, optional backend session sync; future-ready for server KPIs via a single adapter.

Implementation Implications for Developers

- UI/Frontend

- Maintain modular ES modules: core/auth, core/storage, modules/users, achievements, tournaments, schedules, tournaments-admin, dashboards.
- Adopt Stitch AI Tailwind tokens across new pages; preserve glassmorphic patterns and accessibility (landmarks, ARIA, focus-visible).
- Reports page: keep KPI keys stable; any shift to server-side computations should route through a single adapter function without altering UI components.

- Backend
- Prisma User schema is the single source of truth for identity and profile. Extend schema carefully when adding new server-owned entities (e.g., tournaments, registrations).
- Keep cookie session (SameSite=Lax) and validation (mobile = 10 digits). Ensure DOB ISO handling is consistent.
- When expanding backend scope, retain the bridge until legacy modules are migrated; define clear “ownership” per entity to avoid drift.

- Data Ownership and Migration
- Users: backend-owned; mirrored to local.
- Achievements, Tournaments, Schedules, Registrations: localStorage-owned for now; plan per-entity migrations later.
- No remapping of legacy IDs; treat backend users as fresh.

Code Architecture Modifications

- Introduced API client (public/js/api.js) with register/login/logout/me/updateMe.
- Backend mirror function (syncLocalSession) updates/creates local user, sets currentUser, preserves achievements and registeredTournaments arrays.
- New Reports modules:
- js/modules/reports.store.js — computes KPIs from localStorage; subscribable state.
- js/modules/reports.ui.js — renders KPI values, manages skeletons and micro-interactions.
- js/reports.js — page controller: guard, chip interactions, lazy loading, optional backend seeding via API.me.
- CSS/Design: Continued use of global glassmorphic components; Tailwind tokens (primary #607afb, background-dark #0f1323), Inter font, rounded-xl; reduced-motion support.

New Feature Specifications

- Government Verification
- Actions only within official’s region; browse-any-region allowed; chips:

Pending=SUBMITTED, Approved, Rejected.
- Admin Tournament Lifecycle
- Govt approval mandatory before publish; unpublish hides from Player Find immediately; Admin retains registrant management; planned player notifications (future).
- Enforce registration.lastDate; disallow duplicate registrations per player per tournament.
- Coach Reports (Landing)
- KPIs: AttendanceRate(30D), Achievements(Approved vs Pending), ActivePlayers(7D), UpcomingSessions(7D), Registrations(PENDING vs CONFIRMED).
- UI: sticky header and filter bar (placeholders), responsive grid, skeletons, deep-link placeholders for drilldowns.
- Data: local-first; optional backend session seeding via API.me; adapter-ready for future /reports API.

Change Impact Evaluation

- Technical Debt
- Hybrid data ownership (Users in DB; others in local) increases complexity and risk of divergence. Mitigate with consistent bridge sync and a clear migration roadmap.
- Allowing any proof file type without a file storage strategy burdens localStorage and complicates previews. Plan MIME-aware handling and server storage in later phases.

- Performance Implications
- Base64 assets in localStorage can approach the 5–10 MB browser quota; compress or externalize in future backend phases.
- Client-side aggregations (Reports) are fast at small scale but will become costly with growth; the /reports server endpoint should consolidate aggregates.

- Architectural Compatibility
- Cookie-based same-origin session is compatible with current static serving; CSRF mitigations required if moving to cross-origin frontend.
- Prisma with SQLite is fit for dev; plan for MySQL/Postgres and file storage (S3/R2) in production.

- Resource Requirements
- Backend maintenance (Prisma migrations, health monitoring) and minimal devops for CodeSandbox or equivalent.
- QA across role guards, policy enforcement (deadline, duplicates), and regression tests for legacy modules after bridge sync.

Actionable Items (For AI-driven Development)

1. Stabilize Policies in Code
- Enforce: Govt approval required before publish, duplicate registration block, lastDate cutoff, region-limited actions.

2. Reports Adapter Introduction (Optional Next)
- Add a single adapter API (getCoachReports) that tries GET /api/reports/coach then falls back to local compute; keep KPI keys identical.

3. UI Hardening
- Achievements: handle non-image proofs (filename/MIME display; no preview).
- Reports: maintain a11y labels and reduced-motion fallback; test breakpoints.

4. Backend Roadmap
- Define ownership and endpoints for tournaments, registrations, achievements, schedules to retire localStorage gradually.
- Plan storage for media (avatars, proofs) and migrate from base64 to URLs.

5. QA and Observability
- Add test data seed scripts for schedules/scheduleRequests/registrations to validate Reports KPIs.
- Add minimal server health checks (GET /api/health already exists) and logs for session sync failures.

Appendix A — Current Backend Endpoints

- POST /api/auth/register
- POST /api/auth/login
- POST /api/auth/logout
- GET /api/me
- PATCH/api/me
- GET /api/health

Appendix B — Key Data Structures (Local)

- sportiqData: users[], currentUser, schedules[], scheduleRequests[], tournaments[]
- User: achievements[], registeredTournaments[] (snapshots)
- Tournament: lifecycle DRAFT !’ SUBMITTED !’ APPROVED !’ PUBLISHED/REJECTED
- ScheduleRequest: PENDING/APPROVED/REJECTED
- Registration: regStatus PENDING/CONFIRMED/REJECTED (missing !’ CONFIRMED)

This document captures the definitive changes and their implications for the responsive UI and backend systems. It is intended to serve as a handoff-ready reference for future development phases and AI-assisted implementation.