HockeyStatsTS

Detailed Engineering Implementation Plan **Date:** January 2026

Target Stack: React, TypeScript, Firebase, Redux Toolkit

Contents

Part I

Phase 0: Environment & Boilerplate

Goal: A running "Hello World" app with correct folder structure and tooling.

0.1 Project Initialization

- ☐ **Scaffold Project:** Run `npm create vite@latest hockeystats - -template react-ts`.
- ☐ **Clean Up:** Remove `src/assets`, `src/App.css`, and clear `src/index.css`.
- ☐ **Git Init:** Initialize repository and create `.gitignore`.

0.2 Dependency Installation

- ☐ **Core Logic:** `npm install inversify reflect-metadata date-fns zod clsx tailwind-merge`
- ☐ **State Management:** `npm install @reduxjs/toolkit react-redux redux-persist`
- ☐ **Routing:** `npm install react-router-dom`
- ☐ **Backend:** `npm install firebase`
- ☐ **Styling (Dev):** `npm install -D tailwindcss postcss autoprefixer`
- ☐ **UI Assets:** `npm install lucide-react`

0.3 Configuration

- ☐ **TypeScript Config:** Update `tsconfig.json`:
  - Set `"strict": true`
  - Set `"noImplicitAny": true`
  - Enable decorators: `"experimentalDecorators": true`, `"emitDecoratorMetadata": true`
- ☐ **Tailwind Init:** Run `npx tailwindcss init -p`. Configure `content` paths in `tailwind.config.js`.
- ☐ **Global Styles:** Add Tailwind directives (@tailwind base; ...) to `src/index.css`.

0.4 Folder Structure Setup

- ☐ Create `src/core/types`, `src/core/di`, `src/core/schemas`, `src/core/constants`
- ☐ Create `src/features/auth`, `src/features/games`, `src/features/roster`, `src/features/stats`
- ☐ Create `src/infrastructure/firebase`, `src/infrastructure/repositories`
- ☐ Create `src/shared/components`, `src/shared/layouts`, `src/shared/hooks`

0.5 Asset Migration

- ☐ Copy `icerink_up.jpg` and team logos from old project to `public/` folder.

Part II

Phase 1: The Domain Layer (Contracts)

Goal: Define all Data Types. Pure TypeScript, no UI.

0.6 Constants & Enums

☐ Create `src/core/constants.ts`.

☐ Export Enums: `GameStatus`, `ActionType`, `Period`, `PlayerPosition`, `PenaltyType`.

0.7 Entity Interfaces

☐ Create `src/core/types/entities.ts`.

☐ Define interfaces: `Player`, `Team`, `GameRosterEntry`.

0.8 Game & Action Interfaces

☐ Create `src/core/types/actions.ts`.

☐ Define Polymorphic types: `BaseAction`, `GoalAction`, `PenaltyAction`, `GameAction` (Union).

☐ Create `src/core/types/game.ts`. define `Game` aggregate interface.

0.9 Service Interfaces (Abstractions)

☐ Create `src/core/interfaces/services.ts`.

☐ Define `IGameService`: `createGame`, `addAction`, `finalizeGame`.

☐ Define `IAuthService`: `login`, `logout`, `getCurrentUser`.

0.10 Validation Schemas

☐ Create `src/core/schemas/game.schema.ts`.

☐ Implement Zod schemas matching the interfaces above for runtime validation.

Part III

Phase 2: Infrastructure & Services

Goal: Connect to Firebase and set up Dependency Injection.

0.11 Firebase Setup

- ☐ Create `src/infrastructure/firebase/config.ts`: Initialize App, Auth, Firestore.
- ☐ Create `src/infrastructure/firebase/converters.ts`: Implement Firestore Data Converters for Type Safety.

0.12 Dependency Injection

- ☐ Create `src/core/di/types.ts`: Define Symbols (`TYPES.IGameService`).
- ☐ Create `src/core/di/container.ts`: Configure Inversify container.

0.13 Auth Implementation

- ☐ Implement `FirebaseAuthService` in `src/infrastructure/services/`.
- ☐ Implement `loginWithGoogle`: Trigger popup, check whitelist collection.

0.14 Game Service Implementation

- ☐ Implement `FirestoreGameService`.
- ☐ Method `createGame`: Omit ID, save to `games` collection.
- ☐ Method `addAction`: Use `arrayUnion` to append to `events` array.
- ☐ Method `updateAction`: Read document, find index, update array (Transaction required).

Part IV

Phase 3: State Management (Redux)

Goal: Handle data flow and Offline Persistence.

0.15 Store Configuration

- ☐ Create `src/store/index.ts`.
- ☐ Configure `redux-persist` with `storage` (localStorage).
- ☐ Define `RootState` and `AppDispatch` types.

0.16 Auth Slice

- ☐ Create `src/features/auth/authSlice.ts`.
- ☐ State: `user` (Serialized), `status`, `error`.
- ☐ Action: `setUser` (called by Firebase `onAuthStateChanged`).

0.17 Games Slice (Entity Adapter)

- ☐ Create `src/features/games/gamesSlice.ts`.
- ☐ Use `createEntityAdapter<Game>` to manage normalized state.
- ☐ Thunk `syncGame`: Subscribe to Firestore `onSnapshot`.
- ☐ Thunk `saveLocalAction`: Optimistic update local state -> Call Service.

0.18 Roster Slice

- ☐ Create `src/features/roster/rosterSlice.ts`.
- ☐ Thunks to fetch all Teams and Players (cached).

Part V

Phase 4: UI Foundation & Routing

Goal: A working visual shell.

0.19 Layouts

- ☐ Create `src/shared/layouts/RootLayout.tsx`: Sidebar navigation + Outlet.
- ☐ Create `src/shared/components/Sidebar.tsx`: Nav links.

0.20 Components (Design System)

- ☐ Create `Button.tsx`, `Input.tsx`, `Card.tsx` using Tailwind.
- ☐ Create `Modal.tsx`: Base component with Backdrop and Focus Trap.

0.21 Routing

- ☐ Create `src/shared/components/ProtectedRoute.tsx`: Redirect to login if auth slice is empty.
- ☐ Configure `App.tsx`:
  - Route `/login` (Public)
  - Route `/` (Private, Home)
  - Route `/games/new` (Private)
  - Route `/games/:id/log` (Private)

Part VI

Phase 5: Master Data & Setup

Goal: Allow creating the data needed to play a game.

0.22 Login Page

- ☐ Create `src/features/auth/LoginPage.tsx`.
- ☐ Add Google Sign-In Button calling `authService.login()`.
- ☐ Handle Errors (Whitelist rejection).

0.23 Admin Dashboard (Rosters)

- ☐ Create `src/features/roster/pages/TeamListPage.tsx`.
- ☐ Create `TeamForm.tsx`: Name, ShortName, Logo Upload.
- ☐ *Optional:* Create script to import JSON data from old project to Firestore.

0.24 Game Setup Wizard

- ☐ Create `src/features/games/pages/NewGamePage.tsx`.
- ☐ **Step 1:** Form for Season, Championship, Date, Home/Away Teams.
- ☐ **Step 2:** `RosterSelector` component. Lists players, Checkbox for "Dressing".
- ☐ **Step 3:** Submit -> Call `gameService.createGame()` -> Redirect to Log.

Part VII

Phase 6: The Game Logger (Core)

Goal: The Interactive Rink and Event Engine.

0.25 Layout

- ☐ Create `src/features/game-logging/GameLogLayout.tsx`.
- ☐ Implement Grid: Header (Score), Main (Rink), Sidebar (Feed).
- ☐ **Mobile:** Implement Split View (Fixed Top, Scrollable Bottom).

0.26 Interactive Rink

- ☐ Create `InteractiveRink.tsx`.
- ☐ Render `icerink_up.jpg`.
- ☐ `onClick` handler: Calculate % coordinates relative to image dimensions.
- ☐ Render Dots: Map over `game.events` and render absolute positioned divs.

0.27 Action Flow (State Machine)

- ☐ Create `useGameLogger` hook to manage flow state: `IDLE` -> `SELECT_TYPE` -> `SELECT_PLAYER` -> `CONFIRM`.
- ☐ **Modal 1:** `ActionTypeSelector`. Grid of Icons.
- ☐ **Modal 2:** `PlayerSelector`. List of Roster entries. Add "Unknown Player" option.
- ☐ **Modal 3:** `ConfirmationModal`. Show summary. Buttons: Confirm/Edit/Cancel.

0.28 Logic Integration

- ☐ On Confirm: Dispatch `saveLocalAction`.
- ☐ Update `gamesSlice` to optimistic add action.
- ☐ Service syncs to Firestore in background.

Part VIII

Phase 7: Post-Game & Analytics

Goal: Review workflow and Stats calculation.

0.29 Staging Workflow

- ☐ Add "Finish Game" button to Logger. Updates status to `STAGING`.
- ☐ Create `AdminGameList.tsx`: Filter by `status == STAGING`.

0.30 Review Interface

- ☐ Reuse `GameLogLayout`.
- ☐ Enable "Edit Mode": Clicking a dot opens `EditEventModal` instead of creating new.
- ☐ Add "Finalize" button (Admin only).

0.31 Stats Engine

- ☐ Create `src/features/stats/engine.ts`.
- ☐ Implement `calculateBoxScore(game)`.
- ☐ Implement `calculateGoalieStats(game)`.

0.32 Stats View

- ☐ Create `GameStatsPage.tsx`.
- ☐ Render "Box Score" table.
- ☐ Integrate `Recharts` for Shot Map visualization.

Part IX

Phase 8: Polish & Deployment

Goal: Production Ready.

0.33 Testing

☐ **E2E:** Install Playwright. Write test: Login -> Create Game -> Log Goal -> Save.

☐ **Unit:** Test `stats/engine.ts` with complex game scenarios.

0.34 Security

☐ Create `firestore.rules`.

☐ Logic: `allow write:  if exists(/databases/$(database)/documents/whitelisted_users/$(reques`

0.35 Deployment

☐ Run `npm run build`.

☐ Initialize Hosting: `firebase init hosting`.

☐ Deploy: `firebase deploy`.