🗂️ PROJECT_STRUCTURE.md — Monorepo Map & Conventions

Repository: `music-community-platform`
Last Updated: November 12, 2025

📦 Directory Tree (top-level)

```
apps/
  web/                    # Next.js 15 (Vercel)
  api/                    # NestJS (Fly.io / App Runner)
  socket/                 # Socket.IO (Fly.io / App Runner)
  workers/
    transcoder/           # FFmpeg worker (AWS Fargate / Fly.io)
packages/
  ui/                     # Shared UI components (Tailwind + shadcn/ui)
  types/                  # Zod schemas; emits OpenAPI
  sdk/                    # Generated API/Socket clients from OpenAPI
infra/
  prisma/                 # Prisma schema + PostGIS migrations + seeds

.docs/ (generated)
```

🧩 Package Conventions

- **TypeScript**: strict mode everywhere;
- **Imports**: use path aliases via `tsconfig.base.json` (e.g., `@uprise/ui`, `@uprise/types`, `@uprise/sdk`).
- **Build**: Turborepo tasks — `build`, `lint`, `typecheck`, `test` across workspaces.
- **Testing**: Jest/Vitest per app; Playwright for web.
- **OpenAPI**: `packages/types` generates `openapi.json`; `packages/sdk` regenerates typed clients. CI fails on drift.

🔐 Web-Tier Contract (enforced by CI)

Overview

The UPRISE platform enforces strict architectural boundaries between the web tier and backend services. This separation ensures security, maintainability, and proper separation of concerns.

Core Principles

1. **No Direct Database Access from `apps/web`**
   - Web tier must never import `@prisma/client` or any database drivers
   - All data access goes through `apps/api` REST endpoints

   - This prevents database credentials from being exposed in client bundles
   - Reduces bundle size and improves web performance

2. **No Server-Side Secrets in Client Components**
   - Client components must never access non-`NEXT_PUBLIC_` environment variables
   - Server-side secrets (JWT_SECRET, DATABASE_URL, AWS keys) are strictly prohibited
   - Use `NEXT_PUBLIC_` prefix for client-safe variables only

3. **No Direct Server Imports**
   - Web tier must not import directly from `apps/api/src` or `apps/socket/src`
   - Use shared packages (`@uprise/types` , `@uprise/ui`) for common code
   - Use API client (`@/lib/api`) for backend communication
   - Use Socket.IO client (`@/lib/socket`) for real-time features

4. **All Mutations Through API**
   - Server actions in `app/` routes should be read-only or delegating
   - Write operations must go through `apps/api` endpoints
   - Real-time features are subscribe-only via `apps/socket`

Enforcement Mechanism

The Web-Tier Contract is enforced through multiple layers:

1. Automated CI Check (T5 Implementation)

- Script: `scripts/infra-policy-check.ts`
- Runs on every PR and push to `main` / `develop`
- Scans all TypeScript/JavaScript files in `apps/web`
- Detects prohibited patterns and fails build on violations

2. Runtime Guards (T1 Implementation)

- File: `apps/web/src/lib/web-tier-guard.ts`
- Throws errors if prohibited code executes
- Provides `assertNotWebTier()` , `guardPrismaClient()` , `guardDatabaseAccess()`

3. ESLint Rules

- File: `apps/web/.eslintrc.json`
- `no-restricted-imports` rule blocks prohibited modules
- Provides immediate feedback during development

4. TypeScript Types

- File: `apps/web/src/lib/types/web-tier.d.ts`
- `ApiOnly<T>` type marker for API-tier only values
- `WebSafe<T>` type marker for web-safe values
- Prisma Client declared as `never` type in web context

Prohibited Patterns

Category	Examples	Reason
Database	`@prisma/client`, `pg`, `mongodb`, `mongoose`	Security, bundle size, architecture
Server Imports	`apps/api/src/*`, `apps/socket/src/*`	Tight coupling, circular dependencies
Secrets	`DATABASE_URL`, `JWT_SECRET`, `AWS_SECRET_ACCESS_KEY`	Security breach risk
AWS SDK	`aws-sdk`, `@aws-sdk/*`	Server-side only, bundle bloat
File System	`fs`, `node:fs`	Not available in browser
Server Modules	`child_process`, `node:child_process`	Server-only, security

Allowed Patterns

Pattern	Purpose	Example
API Client	Backend communication	`import { api } from '@/lib/api'`
Socket Client	Real-time features	`import { io } from 'socket.io-client'`
Shared Packages	Common types/UI	`import { Button } from '@uprise/ui'`
Public Env Vars	Client-safe config	`process.env.NEXT_PUBLIC_API_URL`
Client Utils	Web-tier helpers	`import { formatDate } from '@/lib/utils'`

Example: Correct vs Incorrect Usage

❌ **INCORRECT:**

```tsx
// apps/web/src/app/users/page.tsx
import { PrismaClient } from '@prisma/client';

export default async function UsersPage() {
  const prisma = new PrismaClient();
  const users = await prisma.user.findMany();
  return <div>{users.map(u => u.name)}</div>;
}
```

✅ **CORRECT:**

```tsx
// apps/web/src/app/users/page.tsx
import { api } from '@/lib/api';

export default async function UsersPage() {
  const users = await api.get('/users');
  return <div>{users.map(u => u.name)}</div>;
}
```

❌ **INCORRECT:**

```ts
// apps/web/src/lib/config.ts
export const dbUrl = process.env.DATABASE_URL;
export const jwtSecret = process.env.JWT_SECRET;
```

✅ **CORRECT:**

```ts
// apps/web/src/lib/config.ts
export const apiUrl = process.env.NEXT_PUBLIC_API_URL;
export const socketUrl = process.env.NEXT_PUBLIC_SOCKET_URL;
```

Related Documentation

- `RUNBOOK.md` (./RUNBOOK.md) - Web-Tier Contract Guard section
- `STRATEGY_CRITICAL_INFRA_NOTE.md` (./STRATEGY_CRITICAL_INFRA_NOTE.md) - Infrastructure policy
- `apps/web/WEB_TIER_BOUNDARY.md` (../apps/web/WEB_TIER_BOUNDARY.md) - Detailed web-tier documentation

Running the Contract Guard

```bash
# Run locally
pnpm run infra-policy-check

# Show help
pnpm run infra-policy-check --help

# Verbose output
pnpm run infra-policy-check --verbose
```

🧵 Branching & PR Rules

- Default branch: `main`
- Feature branches: `feat/<scope>` ; chore/bugfix similarly
- Every PR:
- includes `Deployment Target:` and `Phase:` lines
- links to relevant spec(s) under `/docs/Specifications`
- updates `CHANGELOG.md`

📚 Specs Index

Specs live under `/docs/Specifications`. Core files:
- `01_UPRISE_Master_Overview.md`
- `02_UPRISE_Skeleton_Framework.md`
- `04_UPRISE_Community_Location_System.md`
- `06_UPRISE_Song_Management_System.md`
- `07_UPRISE_Discovery_Map_System.md`
- `08_UPRISE_Events_System.md`
- `09_UPRISE_Promotions_Business.md`

> If migrating specs from the legacy repo, preserve filenames and IDs so cross-references remain valid.

🧰 Scripts (common)

```
pnpm -r dev        # run all apps in dev mode
pnpm -r build      # build all workspaces
pnpm -r test       # run tests
pnpm prisma migrate dev   # dev migrations
pnpm prisma migrate deploy # prod migrations
```

🧯 Incident Response (quick)

1. Check Sentry for errors (web/api) and logs (workers/socket).
2. Roll back last deployment on target platform (Vercel/Fly/AWS) if needed.
3. Open a `hotfix/*` branch; patch and ship.