Verity App Summary (Repo-Based)

What It Is

Verity is a real-time matching and short video session platform with double opt-in matching before identity reveal and chat.

The repository contains a NestJS backend plus React web and React Native mobile clients, with Australia-first deployment guidance.

Who It Is For

Explicit primary user/persona statement: Not found in repo.

Inferred from implemented flows: end users seeking brief live introductions before choosing MATCH or PASS.

What It Does

- Anonymous onboarding/auth with JWT access plus rotating refresh tokens; optional phone/email verification endpoints.
- Token-gated queue join/leave using Redis, including refunds on leave when no match is made.
- Background matching worker pairs queued users and creates timed sessions.
- 45-second live video sessions using server-issued Agora RTC/RTM tokens and server-authoritative session end events.
- Double opt-in decisions (MATCH/PASS) create mutual matches and unlock identity reveal and chat only on mutual match.
- Persistent and real-time chat for match participants using PostgreSQL storage and Socket.IO events.
- Payments (Stripe checkout/webhook), moderation (Hive webhook plus report/block), analytics, notifications, and feature flags.

How It Works (Architecture)

- Clients: `verity-web` (Vite/React) and `verity-mobile` (React Native) call backend REST APIs and Socket.IO namespaces (`/queue`, `/video`, `/chat`).
- Backend: NestJS app composes modules for auth, queue, session, video, chat, payments, moderation, notifications, monitoring, analytics, and flags.
- State/data: Prisma persists users, sessions, matches, messages, transactions, and moderation records in PostgreSQL; Redis stores queue state, locks, and timed session metadata.
- Flow: join queue (token debit) -> worker match -> session + video tokens -> choices -> mutual match/non-mutual -> chat and notifications.
- Deployment evidence: same backend image can run API and dedicated worker, distinguished by `ENABLE_MATCHING_WORKER`.

How To Run (Minimal)

1. At repo root: `npm ci`, then configure environment using `.env.production.example` (minimum local essentials include `DATABASE_URL`, `REDIS_URL`, and JWT secrets).
2. Bring up PostgreSQL and Redis, then sync schema for local dev: `DATABASE_URL=... npx prisma db push --accept-data-loss`.
3. Start backend API: `npm run start:dev` (defaults to port 3000).
4. Start web client: `cd verity-web && npm install && npm run dev` (set `VITE_API_URL`, `VITE_WS_URL`, and `VITE_AGORA_APP_ID`).
5. Mobile local startup commands: Not found in repo.

Evidence sources: README.md, src/app.module.ts, src/main.ts, prisma/schema.prisma, verity-web/README.md, verity-mobile/README.md