**ThrivioHR — End-to-End Development Plan (Keep-Forever Version)**

Rule of the road: we **do not** move to the next step until the previous step’s **acceptance checks** are green and stable. Every task ships with a **surgical Agent prompt** and a **Definition of Done (DoD)**.

**Big 0 — Operating Principles (how we work)**

**Technical:** Single repo (pnpm+Nx+TS), strict linting/format, 100% coverage on core domains, ephemeral DB tests, reproducible prompts.

**Plain English:** We build in tiny, testable slices. Each slice has a checklist. We only proceed when green.

**DoD (global):**

* Lint clean, tests green, build passes, CI passes on PR.
* Docs updated (roadmap + any touched READMEs).
* No “TODO later” debt; if deferred, tracked in /docs/adr or /docs/todo.md.

**Surgical Prompt Template (pinned)**

vbnet

CopyEdit

Execute this EXACT task. Do not touch anything beyond what’s listed.

If any command fails, PRINT the error and STOP. Do not “fix” or scaffold.

GOAL

<1–2 lines>

TOUCH ONLY

<explicit files>

CHANGES

<precise edits or full replacements>

RUN & PRINT

<commands + what to print>

STOP.

**Big 1 — Infrastructure & Repo (Foundations)**

**1.1 Monorepo + Tooling**

**Technical:** pnpm+Nx workspace, ESLint w/ decorators, TypeScript strict, shared packages.

**Plain English:** Repo is organized, consistent code style, ready for teams.

**DoD:** pnpm -w lint + pnpm -w build pass locally.

**Prompt (already executed):** Init monorepo, add base apps/services, ESLint config.

**1.2 Managed Postgres + Secrets**

**Technical:** Replit Managed PG (Neon), DATABASE\_URL in secrets, healthcheck.

**Plain English:** We have a real database ready for dev/tests.

**DoD:** pnpm db:check prints DB OK.

**1.3 CI Quickstart**

**Technical:** GitHub Actions: lint → identity tests → build; Postgres service; nx caches.

**Plain English:** Every push/PR runs checks in the cloud automatically.

**DoD:** CI passes on PR to main.

**Big 2 — Database & Migrations (Dev/Test Strategy)**

**2.1 Drizzle ORM + Migrations**

**Technical:** Drizzle schemas, drizzle out folder, journal committed.

**Plain English:** DB structure is code-first and versioned.

**DoD:** pnpm db:generate && pnpm db:migrate succeeds.

**2.2 Ephemeral Test Schemas**

**Technical:** Jest runs **manual SQL migration runner** that strips "public". to bind FKs to the current test schema.

**Plain English:** Each test run uses a fresh, isolated schema, so tests don’t trip over each other.

**DoD:** Identity tests green from a clean DB.

**Big 3 — Identity (People, Orgs, Access)**

**3.1 Identity Data Layer**

**Technical:** 10 tables (organizations, users, identities, sessions, roles, role\_bindings, org\_units, org\_membership, locations, employment\_events); constraints & cascades; camel↔snake mapping fixed.

**Plain English:** We can represent orgs, employees, logins, and org charts in the database safely.

**DoD:** 4 test suites green (constraints, cascades, sessions index, schema import). 100% statements/branches/lines on schema.

**Micro-steps / Prompts used:**

* Create schema + migrations
* Add manual SQL runner in tests
* Add deterministic helpers (createOrg → createUser → createSession)
* Fix FK issues; alias userId/id

**3.2 Identity REST (BFF) (🔜 next)**

**Technical:** NestJS module: POST/GET /orgs, POST/GET /users, DTO validation, Drizzle repo.

**Plain English:** Admin tools (and later the UI) can create/list orgs and users via a clean API.

**DoD:**

* npx nx build bff passes
* Local smoke with curl returns JSON objects for orgs/users
* No auth yet; **read/write works** against DB

**Micro-steps / Prompt (ready to paste):**  
I gave you the “Implement Identity REST” prompt in the previous message—use it as Step A in your queue.

**3.3 Sessions/JWT Auth (BFF) (➡ after REST)**

**Technical:** Minimal POST /auth/login { email } issues a session & JWT; guard protects /users.

**Plain English:** You can log in and only see protected routes when authenticated.

**DoD:** curl demo: login → get token → access protected endpoint.

**Micro-steps / Prompt:** I’ll hand it over after 3.2 is merged.

**Big 4 — Directory (Org data ingestion)**

**4.1 CSV/JIT Imports (MVP)**

**Technical:** Ingest endpoints + validation; map to users, org\_units, org\_membership; idempotency.

**Plain English:** HR can upload a CSV or let users appear via SSO (JIT).

**DoD:** Sample CSV imports 50 users & department tree with zero duplicates.

**Micro-steps / Prompt:** Scaffold services/directory schema + tests; REST in BFF.

**4.2 SSO (OIDC + SAML) wiring**

**Technical:** OIDC & SAML providers (authN), JIT user creation; per-tenant config.

**Plain English:** Companies can log in with Okta/Azure AD/Google; new users appear automatically.

**DoD:** Test tenant logs in via OIDC; user appears in users + identities.

**Big 5 — Recognition (Non-monetary first)**

**5.1 Recognition Data Layer (MVP)**

**Technical:** Tables: recognitions, reactions, optional values[]; FK to org/users; constraints & cascades; ephemeral tests.

**Plain English:** Employees can post appreciations; others can react.

**DoD:** pnpm test:recognition green (constraints, cascades, feed order).

**5.2 Recognition REST (BFF)**

**Technical:** POST /recognitions {from,to,message,values}; GET /feed?orgId.

**Plain English:** Create an appreciation and read a feed.

**DoD:** curl demo creates + lists recognitions.

**Big 6 — Budgets & Ledger (Points later)**

**6.1 Ledger Core**

**Technical:** Double-entry tables (wallets, entries, postings); consistent invariants; FX/PMV hooks.

**Plain English:** Money-like points are recorded transparently and immutably.

**DoD:** Unit tests that prove debits=credits; reversal works.

**6.2 Budgets & Allowances**

**Technical:** Tenant → Dept → Team allocations; P2P monthly **by count**; defaults: carryover OFF; overage BLOCK.

**Plain English:** Finance/HR can allocate and cap recognition budgets.

**DoD:** Allocation tests + policy checks pass.

**Big 7 — Marketplace (API-ready, empty catalog)**

**7.1 Marketplace Skeleton**

**Technical:** Catalog, Offers, Orders, Fulfillment; **ProviderAdapter** interface; webhooks; audit.

**Plain English:** Shop shell exists; we can plug providers later without changing core.

**DoD:** Create a “mock card” order; lifecycle moves to fulfilled with a stub adapter.

**Big 8 — Notifications**

**Technical:** Outbox → worker; email/push channels; templates & i18n.

**Plain English:** People actually get notified on recognizes, allocations, invites.

**DoD:** Local dev SMTP prints emails; test proves send & idempotency.

**Big 9 — Insights & AI Foundation**

**Technical:** Event capture; simple rules engine; AIAdapter interface; privacy guardrails.

**Plain English:** “Nudges” like “X hasn’t logged in in 10 days” or “Y took 5 leaves in 20 days”.

**DoD:** 3 sample insights generated from seeded data; endpoint returns them.

**Big 10 — i18n & White-Label**

**Technical:** next-intl/i18next; locale packs (en, fr, es, de, it, nl, ar/RTL); brand theming.

**Plain English:** UI changes language; tenants can brand colors/logo.

**DoD:** Language switcher works; RTL verified; tenant theme applies.

**Big 11 — Security & Observability**

**Technical:** RBAC+ABAC (policies), audit logs, rate limits, OpenTelemetry, Sentry.

**Plain English:** Access is correct, everything is trackable, issues are visible.

**DoD:** Policy tests pass; traces visible locally; error captured.

**Big 12 — Deployment (AWS) & Data Residency**

**Technical:** IaC (CDK/Terraform); ECS/EKS, RDS; regions: EU, UAE+GCC, Morocco; rollout strategy.

**Plain English:** We can deploy to customer-compliant regions.

**DoD:** One region live (staging); smoke tests green.

**Big 13 — Compliance & DR**

**Technical:** Backups, PITR, anonymized dumps; DPIA; incident response doc.

**Plain English:** If something breaks, we can restore; we know our obligations.

**DoD:** Restore drill succeeds; docs in /docs/security/.

**Big 14 — Launch Readiness**

**Technical:** SLOs (availability/latency), error budgets; on-call; runbooks; pricing toggles; entitlements.

**Plain English:** We’re ready to sell, monitor, and support.

**DoD:** Pilot tenant onboarded end-to-end.

**File Map (Docs you can keep alongside)**

You already have many of these. Here are the **key MDs** to pin:

**/context/ROADMAP.md (paste this plan here)**

* This entire document (you can keep it as your master plan).

**/context/prompt-templates/SURGICAL-PROMPT.md**

cpp

CopyEdit

Execute this EXACT task...

<the template from Big 0>

**/docs/DEFINITION\_OF\_DONE.md**

* Global DoD (lint/tests/build/CI/docs), plus per-domain DoD checklists.

**/docs/OPERATIONS.md**

* How to run locally, seed, reset DB, common scripts.

**/docs/SECURITY.md**

* RBAC/ABAC model summary, auth flows, secrets handling, data retention.

**/docs/ADR/0001-... .md**

* Architecture Decision Records for big calls (tenancy model, eventing, AI adapter, etc.).