Max Potential Basketball – Developer Handoff (Supabase + n8n, Modular Drill Engine)

1 Core schema (unchanged across verticals)

------------------------------------------------

actor(id UUID PK, name TEXT, kind TEXT, created\_at TIMESTAMPTZ)

profile(id UUID PK, actor\_id FK actor(id), json\_attrs JSONB, created\_at)

observation(id UUID PK, actor\_id FK, note\_type TEXT, raw\_text TEXT,

tags TEXT[], json\_attrs JSONB, created\_at)

intervention(id UUID PK, actor\_id FK, type TEXT, status TEXT,

start\_date DATE, planned\_end DATE, constraint\_tags TEXT[],

json\_attrs JSONB, created\_at)

metric(id UUID PK, target\_id UUID, metric\_type TEXT, value NUMERIC, unit TEXT, created\_at)

tag(id UUID PK, name TEXT UNIQUE, namespace TEXT, embedding vector(1536))

link(id UUID PK, left\_id UUID, right\_id UUID, rel\_type TEXT, json\_attrs JSONB, created\_at)

2 Basketball overlay tables

------------------------------------------------

overlay\_pdp(intervention\_id UUID PK → intervention(id), pdp\_level INT, focus\_area TEXT)

overlay\_drill(intervention\_id UUID PK → intervention(id),

chaos\_level INT,

suggested\_format TEXT,

base\_drill\_uid TEXT, -- e.g. “DR-001”

constraint\_combo JSONB) -- generated combination record

3 Modular Drill‑Builder logic (Supabase)

------------------------------------------------

\* Base drills sit as static rows in a small reference table (`base\_drill(id,name,base\_tags)`).

\* Postgres FUNCTION `build\_modular\_drill(\_base\_id uuid, \_constraints text[])`:

– Creates a new row in `intervention` (type='Drill', constraint\_tags=\_constraints)

– Inserts a matching row in `overlay\_drill` with

`base\_drill\_uid`, `constraint\_combo := \_constraints::jsonb`

– Returns the new intervention id.

\* Coaches submit constraint tags through an n8n webhook → function call.

Result: unlimited drill variations without new schema.

4 Realtime trigger (copy‑paste)

------------------------------------------------

ALTER PUBLICATION supabase\_realtime ADD TABLE observation;

CREATE OR REPLACE FUNCTION notify\_obs() RETURNS trigger AS $$

BEGIN

PERFORM pg\_notify('observation\_new', row\_to\_json(NEW)::text);

RETURN NEW;

END; $$ LANGUAGE plpgsql;

CREATE TRIGGER obs\_new

AFTER INSERT ON observation

FOR EACH ROW EXECUTE PROCEDURE notify\_obs();

5 n8n flow summary

------------------------------------------------

\* Capture (webhook / shortcut) → INSERT observation.

\* Realtime listener → GPT tag/score → UPDATE intervention / metric.

\* Weekly digest Slack/email: SQL view top unaddressed constraint tags.

\* Build‑Drill webhook → call `build\_modular\_drill()` → returns UID for Session plan.

6 Migration checklist

------------------------------------------------

1 Insert actors, profiles, observations, interventions, metrics from Airtable.

2 Backfill overlay tables (`overlay\_pdp`, `overlay\_drill`).

3 Smoke‑test CTI & Skill‑delta dashboards.

4 Switch automations from Make to n8n (Supabase nodes).

7 Notify channels

------------------------------------------------

Personal vertical → SMS via Twilio

Basketball coaches → Weekly Slack digest

Thresholds sleeved in `ruleset` table (vertical, role, push\_logic JSONB).

8 File locations & credentials

------------------------------------------------

\* SQL DDL scripts: `/scripts/core\_schema.sql`, `/scripts/overlay\_basketball.sql`

\* Supabase project: MAX Potential

\* n8n workspace: https://n8n.cloud/maxpotential

(End of note)