Build Test - Full-Stack MVP Slice (Next.js + Supabase, Financials-Only)

Goal: Deliver a thin vertical slice of our MVP: a small **Supabase** data model, a **Next.js** page deployed to **Vercel** with a chart and a table, plus a tiny "ask a question" box that returns one finance answer from the DB.

- **Timebox:** 3–4 hours (or 90-min paid pairing alternative)
- **Stack:** Next.js (TypeScript), Vercel, Supabase/Postgres, Recharts **or** Highcharts, ShadCN **or** Mantine
- **APIs: No external APIs required** (financials-only)
- **AI:** Optional. You may implement the "ask" box without an LLM. If you choose to use an LLM, that's fine—but not required.
- **What we're testing:** judgment, speed, correctness, clarity, and minimal polish

1) Requirements (non-negotiables)

A) Data Layer (Supabase)

Create a minimal schema to show **revenue over time** for a team.

Suggested tables

- `teams (id uuid pk, name text unique not null)`
- `financials (team_id uuid fk -> teams.id, season_year int, revenue numeric, ebitda numeric, primary key (team_id, season_year))`

Seed data

- Insert **1–3 teams**.
- Insert **≥3 seasons per team** with `revenue` values (simple numbers are fine).
- Include `ebitda` (even if not charted).

Basic RLS (demo-level)

- Add a brief note and **one simple policy** (e.g., allow read to all authenticated users) or describe a per-user "followed teams" approach. This is about your awareness, not security hardening.

Deliverables

- SQL migration(s) or supabase migration files
- Short note in README explaining your RLS choice

B) UI (Next.js + Vercel)

Build a page that includes:

- A **team selector** (dropdown)
- A **revenue-over-time chart** for the selected team (Recharts or Highcharts)
- A **metrics table** with at least: *season_year*, *revenue*, and **YoY change** (compute either server- or client-side)

UX basics we expect

- Loading and empty states
- Clean, minimal styling using **ShadCN** or **Mantine**
- Deployed preview on **Vercel**

C) "Ask" Box (AI optional, DB-first)

Add a tiny input that answers:

> "What was **Team X's** revenue in **2022**, and what was the **YoY change**?"

Two acceptable implementations

- **No-LLM:** A server route reads DB values and returns a plain, formatted answer (preferred for speed).
- **With LLM (optional):** Server route reads DB → passes a structured result to an LLM → returns a concise answer. (Please do **not** let an LLM execute arbitrary SQL.)

Answer must include: number, currency assumption (e.g., GBP or "units"), and YoY %.

2) Nice-to-Have (financials-only, no external APIs)

Pick **one**. These are tiny extras designed to fit the 3-hour window:

1. **Currency formatting & toggle (10–15 min)**
   Fixed demo FX rate; toggle between GBP/EUR; reformat table and chart.
2. **CAGR badge (10–15 min)**
   Compute revenue **CAGR** across shown seasons; display a small badge (hide if insufficient data).
3. **Two-team comparison (15–20 min)**
   Select and render **two teams** on the same revenue chart with a legend.
4. **Sortable table + CSV download (10–20 min)**
   Sort by season/revenue/YoY and **download CSV** of current rows (client-side).

5. **Sparkline in table (10–15 min)**
   Tiny inline sparkline of revenue trend for the selected team.
6. **Unit test for metrics util (10–15 min)**
   Test **YoY** and **CAGR** edge cases (missing prior year, zero).
7. **States & accessibility polish (10–15 min)**
   Clear empty/error states; keyboard-friendly selector; visible focus.

Tip: One clean bonus > several half-finished. Note any bonus in the README.

3) Acceptance Criteria

1. **Data correctness:** Selecting a team shows ≥3 seasons; YoY matches `financials`.
2. **Chart renders:** Revenue line updates when a different team is selected.
3. **Ask box works:** Returns the correct revenue & YoY for Team X (2022) using DB values (LLM optional).
4. **Deployment:** Working **Vercel preview URL**.
5. **RLS note:** README explains your minimal policy / approach.
6. **Professionalism:** Tidy UI; no secrets in repo; clear README.

4) What to Submit

- **GitHub repo** (TypeScript)
- **Vercel preview URL**
- **Migrations** (`/supabase/migrations` or `.sql`) + brief RLS note
- **`.env.example`** (placeholders only)
- **README** (~1–2 pages) covering:
  - How to run locally & deploy
  - Data model overview + YoY calculation
  - “Ask” box implementation (no-LLM or LLM)
  - What you cut (and why), if you ran out of time
  - How you used AI tools (if at all)
- **(Optional)** A simple time breakdown or Clockify/Toggl screenshot