I wanted to briefly explain *why* we've chosen this specific stack for Nanny Whisperer, so the design decisions make sense from a dev point of view.

---

1. Airtable – source of truth for matching & ops

We're using **Airtable** as the main operational database for a few reasons:

- **Structured but flexible:** We have quite a rich data model for hosts and nannies (location, availability, age groups, skills, lifestyle/values, salary, etc.) plus a 100-point matching algorithm with must-match filters and weighted scores. Airtable lets us model this with tables + formula fields instead of constantly changing SQL schemas.
- **Non-dev friendly UI:** Admin/staff need to browse nannies, filter, and manually review matches/shortlists without dev work. Airtable Interfaces give them a "mini backoffice" with no code.
- **Good enough as a DB for v1:** It's easy to work with over API (simple REST), perfect for a v1 where relational complexity is moderate and we care more about speed than about deep DB tuning.
- **Fast iteration on scoring:** Our match logic (40 core, 20 skills, 20 values, 20 bonus) will evolve. Using Airtable formulas/fields lets us tweak weights and rules without redeploying backend code.

Your interaction with Airtable:

- Read/write: Hosts, Nannies, Shortlists, Matches, InterviewRequests.
- Use it as the truth for shortlist composition, Proceed/Pass state, and scheduled interviews.

---

2. GHL (GoHighLevel) – CRM, payments, comms, calendars

We're using **GHL** as "the brain" for anything CRM-ish:

- **Onboarding forms** already mapped to all the fields in the onboarding docs (host + nanny).
- **Payments & tiers**: €20 membership, Fast Track €500, VIP €3,000, and the €200 contract upsell.
- **Pipelines & workflows**: who is Standard / FT / VIP, when to send emails/SMS, when to issue shortlist links, when to remind about calls, etc.
- **Calendars**:
    - Host calendars for FT & VIP (they choose 5 time slots from here).

    - Kayley's concierge calendar for VIP overlap checks.

- **Conversations**: Standard text chat lives here, no need to build chat yourself.

Your role with GHL:

- Consume webhooks (e.g., "new host onboarded", "payment success", "Proceed/Pass clicked").
- Call back into GHL (or an integration layer) when: meetings are scheduled, status changes, etc.

---

3. Next.js app – the "discreet frontend"

We don't want public search or a big monolith — just a thin, secure UI layer:

- **Shortlist page**: several nanny cards for a given host, accessed via token link.
- **CV page**: full nanny profile (CV-style, classy), also tokenized.
- **Interview request page**: nanny-facing page with host summary + 5 time slots to choose from.

Why Next.js:

- Great for **secure, server-rendered token pages** (SSR or RSC) with noindex.
- Easy to deploy on Vercel, environment variables, etc.
- Nice developer ergonomics for integrating Airtable + Zoom APIs.

---

4. Zoom (single video platform for FT & VIP)

We standardize on **Zoom** for both Fast Track and VIP to avoid fragmentation:

- **Fast Track:**
    - Host picks 5 slots (GHL calendar) → nanny chooses → we auto-create an NW-owned Zoom meeting → no concierge.
- **VIP:**
    - Host picks 5 slots → we intersect with Kayley's calendar → nanny picks from overlapping slots → we create an NW Zoom meeting with concierge as host.

Why Zoom:

- Well-documented API, easy to automate meeting creation.

- Waiting rooms, name control, and chat options → needed for privacy and bypass prevention.
- One integration instead of juggling multiple video providers.

Your part:

- Build the scheduling logic (5-slot selection, nanny choice, retry loop).
- Integrate Zoom's API for meeting creation and push the join links back through GHL workflows.

---

5. Overall design philosophy

- **GHL** does everything that looks like CRM/comms/payments/scheduling.
- **Airtable** holds structured data + scoring + ops view.
- **Next.js app** is a secure UI + integration layer between those two.
- **Zoom** is the single video backend.

You don't need to build a full backend-heavy platform; you're mostly creating:

- Token-protected pages,
- A small set of APIs,
- Some scheduling + Zoom orchestration,
- Glue to keep Airtable and GHL in sync.

If anything about this stack feels suboptimal from your perspective (e.g., where logic should live or how we structure events/webhooks), I'd love your opinion on how to make it cleaner while keeping the same user experience.