Consolidated Call Tracking & CRM Architecture Plan

🎯 Core Stack & Component Roles

Component	Tool Choice	Role in the Architecture
Frontend/PWA	**Next.js**	Client-facing app (CRM/Inbox) and hosting for the tracking script.
Database/Auth	**Supabase**	Core transactional data (User Auth, Client IDs, Billing).
Voice & SMS	**Raw Twilio**	Programmable communications engine (number purchase, forwarding, webhooks).
Configuration Layer	**Airtable**	**Centralized, No-Code Config Manager and Business-Facing CRM/Pipeline.** (Replaces JSON files).

Project Requirements Plan (PRP): Twilio Webhook to Airtable

The core goal of this PRP is to create a reliable, high-speed, serverless endpoint that acts as the "brain" for every inbound Twilio call/SMS, handles the business logic, and logs the data correctly into the client's pipeline in Airtable.

I. Next.js API Route: The Brain (/api/twilio-inbound)

This serverless route handles the incoming POST request (the webhook) from Twilio whenever one of your tracking numbers receives a call.[1]

A. Initial Request Handling & Security (Next.js)

Requirement	Implementation Detail	Rationale
Endpoint	pages/api/twilio-inbound.js (or a Route Handler in the App Router).	Provides a secure, serverless environment to handle the webhook.
Authentication	**Validate Twilio Request Signature.** Use the twilio SDK's request validation feature (validateExpressRequest or similar logic) to	Critical security step for any public webhook.

Requirement	Implementation Detail	Rationale
	ensure the request is genuinely from Twilio, not a malicious third party.	
Identify Source Number	Extract the To phone number from the Twilio POST payload.	This is the unique key to look up the client's configuration.

B. Configuration Lookup (Airtable API)

1. **Look Up Client Config:** Use the To phone number to query your **Airtable Client Configuration Table** (as discussed previously). This query is instant and pulls all necessary data.
2. **Required Data:** The query must return:
   - Client ID (for Supabase/Internal reference)
   - Primary Number (the client's actual office number for forwarding)
   - SMS Missed Call Active? (feature toggle)

C. Business Logic & Response (TwiML)

Based on the nature of the Twilio webhook (Voice or SMS), the route executes the following:

If Twilio Webhook is...	Business Logic	Twilio Response (TwiML)
Inbound Call	1. **Log Call:** Create a new record in the Airtable Lead Pipeline (Table 2) with the caller's number, timestamp, and the Twilio tracking number. 2. **Execute Forward:** Instruct Twilio to connect the call.	Respond with **TwiML** to <Dial> the Primary Number fetched from Airtable.
Missed Call Status Callback	1. **Check Criteria:** Is SMS Missed Call Active? **AND** call status is no-answer **AND** duration is < 15s? 2. **Send Text:** If yes, use the Twilio SDK to send the pre-written missed-call SMS to the caller.	Respond with **HTTP 200** (OK) to acknowledge the status update (no TwiML needed).

If Twilio Webhook is...	**Business Logic**	**Twilio Response (TwiML)**
Inbound SMS	1. **Log SMS:** Create a new record in the Airtable Lead Pipeline (Table 2) and record the message body. 2. **Relay:** Forward the SMS content to the client's actual mobile (via Twilio).	Respond with **TwiML** to <Message> the client's number with the caller's text.

II. Supabase & Airtable Synergy

You have two powerful databases. Here's how to manage their roles:

- **Supabase (Postgres):** Should remain your **core transactional database**. Use it for User IDs, Authentication, Billing records, and the actual raw data the Next.js app needs for real-time operation (e.g., the client's *active* logged-in session data). **This is your high-performance data.**
- **Airtable:** This is your **business-facing data layer** (low-volume, high-value). Use it for:
    1. **Configuration Management (Table 1):** The non-technical settings.
    2. **Lead/CRM Pipeline (Table 2):** Where your clients log in to see/manage their leads.

The Next.js Webhook is the only piece of code that should talk to all three: Twilio, Supabase, and Airtable, ensuring a clean data flow.

Will This Be Easy and Robust?

Aspect	**Evaluation**	**Why it's the Best Approach**
Ease of Integration	**Very Easy.** Next.js API Routes are designed to handle webhooks and make external API calls (to Airtable/Supabase). You only need to code this logic once.	You're using an API (Airtable) as an external JSON config, which is far simpler than building a UI for config management.
Robustness	**Extremely Robust.** Twilio webhooks are highly reliable. Next.js is a stable, serverless	The biggest bottleneck (manually changing code for config) is removed. Airtable

Aspect	Evaluation	Why it's the Best Approach
	environment. Supabase (Postgres) is a rock-solid database.	acts as the stable management layer.
Scalability	**High.** To add a new client, you just **buy a Twilio number** and **add one row in Airtable.** No code changes, no redeployment. This is the definition of a scalable, low-overhead platform.	Your time is spent servicing clients, not managing JSON files.

By centralizing configuration in Airtable, you gain the **control of an engineer** but the **low-maintenance ease of a no-code tool** for client setup.