Nemo B2B

SaaS Architecture Review
Voice AI Platform for SMB Appointment Reminders & Call Marketing

February 2026

Executive Summary

Nemo B2B is a voice AI platform enabling small and medium businesses to automate appointment reminder calls and outbound marketing campaigns. The platform consists of three repositories: a Node.js/Express API, a Python LiveKit voice agent, and a Next.js frontend.

The system is **functional for a pilot** but has significant architectural gaps that must be addressed before operating as a production multi-tenant SaaS. The most critical issues are: no billing integration, a shared phone number for all tenants, a single-file API monolith with in-process schedulers, and no job queue.

Current Architecture

System Overview

Component	Tech Stack	Hosting
API (`nemo-b2b-api`)	Node.js, Express, TypeScript	Railway
Voice Agent (`nemo-b2b-livekit`)	Python 3.12, LiveKit Agents, Gemini Realtime	Railway (Docker)
Frontend (`nemo-b2b-web`)	Next.js 16, React 19, Tailwind CSS 4	Vercel
Database	PostgreSQL	Supabase
Auth	Supabase Auth (JWT)	Supabase
Telephony	Telnyx SIP Trunk via LiveKit	LiveKit Cloud + Telnyx

Call Flow (How It Works Today)

1. **Scheduler** (in-process `setInterval`, every 60s) finds appointments due for reminder
2. **API** dispatches a LiveKit agent to a room and creates an outbound SIP call via Telnyx
3. **Voice Agent** (Python) handles the conversation using Google Gemini Realtime (LLM + STT + TTS)
4. Agent uses tools: `confirm_appointment`, `request_reschedule`, `detected_answering_machine`, `end_call`
5. **Transcript** is captured and posted back to API for storage and AI-powered outcome classification

What's Working Well

- **Voice agent architecture** is solid — clean class hierarchy with `BaseCallAgent` → specialized agents, good error handling, silence detection, transcript capture with retry
- **Campaign framework** is well-designed — type-specific templates (re-engagement, review collection, no-show followup), call windows, concurrency limits, cycle frequency
- **Database schema** is well-modeled — proper enums, FK relationships, cascade rules
- **RLS policies** are defined in Supabase migrations (though not leveraged by the API)
- **Frontend** is functional with Supabase auth and a complete dashboard

---

Critical Gaps for SaaS

1. No Billing Integration (Severity: CRITICAL)

Current State: `subscription_tier` (FREE, STARTER, PROFESSIONAL, ENTERPRISE) and `subscription_status` fields exist in the database but are **never read or enforced anywhere in the codebase**. There is zero payment processing — no Stripe, no metering, no limits, no checkout.

Impact: You cannot charge customers. Any user can make unlimited calls on a FREE tier.

Recommendation:

- Integrate **Stripe** for subscription management
- Create a `checkSubscription` middleware that gates calls, campaigns, and customer counts by tier
- Add a `usage_events` table to meter every call (duration, type, cost)
- Implement Stripe webhooks for subscription lifecycle (created, updated, canceled, payment failed)
- Build a customer-facing usage dashboard

Suggested Tier Structure:

	Free	Starter	Professional	Enterprise
Price	$0	$49/mo	$149/mo	Custom
Reminder Calls	20/mo	200/mo	1,000/mo	Unlimited
Campaign Calls	0	50/mo	500/mo	Unlimited
Customers	25	250	Unlimited	Unlimited
Phone Number	Shared	Dedicated	Dedicated	Dedicated + vanity
Team Members	1	3	10	Unlimited
Voice Options	1	All	All + custom	All + custom

---

2. Shared Phone Number (Severity: CRITICAL)

Current State: A single Telnyx phone number (`+18447751555`) is used as the caller ID for ALL businesses. All calls go through one SIP trunk with shared capacity.

Impact: Customers see the same caller ID regardless of which business is calling. If one business's number gets flagged as spam, all businesses are affected. Businesses cannot port their existing numbers.

Recommendation:

- Use the **Telnyx Number Ordering API** to provision a dedicated number per business at signup (or upgrade)
- Create a `b2b_phone_numbers` table linking numbers to businesses
- Route outbound calls through the business's own number
- Support number porting for businesses with existing numbers
- Implement a number pool for free-tier users (rotating shared numbers)

---

3. Single-File API Monolith (Severity: HIGH)

Current State: All 2,751 lines of API logic — routes, middleware, schedulers, WebSocket handlers, AI session management — live in a single file (`src/index.ts`). There are no route modules, controllers, services, or repository layers.

Impact:

- Impossible for a team to work concurrently without merge conflicts
- In-process `setInterval` schedulers mean you can only run **one API instance**
- In-memory state (`geminiSessions` , `telnyxStreams` Maps) prevents horizontal scaling
- Very difficult to test (only 1 unit test file exists)

Recommendation:

Decompose into a modular architecture:

```
src/
  routes/          # Express route definitions
    appointments.ts
    calls.ts
    campaigns.ts
    customers.ts
    templates.ts
    webhooks.ts
  controllers/     # Request handling, validation
  services/        # Business logic
    reminder.service.ts
    campaign.service.ts
    call.service.ts
    billing.service.ts
  middleware/       # Auth, billing checks, rate limiting
  workers/         # Separate process for background jobs
    reminder.worker.ts
    campaign.worker.ts
  lib/             # Shared utilities, DB client
  types/           # TypeScript interfaces
```

---

4. No Job Queue (Severity: HIGH)

Current State:

- Reminder scheduler: `setInterval(60s)` in the API process
- Campaign scheduler: `setInterval(5min)` in the API process
- If the process crashes, no reminders fire until restart
- No retry limits — a permanently failing call retries every minute forever

- No dead-letter queue, no job visibility, no distributed locking

Impact: Unreliable call delivery. Cannot scale horizontally. No way to inspect, retry, or debug failed jobs.

Recommendation:

- Implement **BullMQ + Redis** for job queues
- Separate queues: `reminders`, `campaigns`, `transcripts`, `billing-events`
- Exponential backoff with max retries (e.g., 3 attempts)
- Dead-letter queue for permanently failing jobs
- Bull Board or similar for job monitoring dashboard
- Run workers as separate processes from the API

---

5. Security Gaps (Severity: HIGH)

Issue	Status	Risk
Telnyx webhook signature verification	**Not implemented** (stub only)	Attackers can forge events
Internal API key	Static, shared, no rotation	Compromised key exposes all internal endpoints
CORS	Allows all origins if env var missing	Cross-origin attacks in production
Service role key	Bypasses all RLS	Any API vulnerability exposes all tenant data
Input validation	Ad-hoc `if` checks, no schema validation	Injection risks
Development fallbacks	Auth silently degrades without env vars	Accidental production exposure

Recommendation:

- Implement Telnyx Ed25519 signature verification (the stub is already there)
- Use a validation library (Zod) for all request bodies
- Enforce required env vars at startup — fail fast if missing
- Rotate internal API keys per deployment
- Add CSRF protection
- Implement DNC (Do Not Call) list management for TCPA compliance

---

6. No Team/Role Management (Severity: MEDIUM)

Current State: One `owner_id` per business. No way for a business to have multiple staff members with different permissions.

Recommendation:

- Create a `b2b_team_members` table with roles: `owner`, `admin`, `staff`
- Staff can manage appointments but not billing
- Admins can manage everything except ownership transfer
- Support email invitations for new team members

---

7. No Monitoring or Observability (Severity: MEDIUM)

Current State: `console.log` with timestamps. No structured logging, no error tracking, no metrics, no alerting.

Recommendation:

- **Error tracking**: Sentry for both API and voice agent
- **Structured logging**: JSON format with correlation IDs (pino or winston)
- **Metrics**: Call success rate, latency percentiles, scheduler health, queue depth
- **Alerting**: PagerDuty/Slack alerts for scheduler failures, high error rates, queue backlog
- **Dashboard**: Grafana or Datadog for operational visibility

---

8. No CI/CD Pipeline (Severity: MEDIUM)

Current State: No GitHub Actions, no automated testing, no automated deployment.

Recommendation:

- GitHub Actions workflow: lint → type-check → test → build → deploy
- Branch protection on `main` requiring passing checks
- Automated deployment to Railway on merge to `main`
- Preview deployments for PRs (Vercel already does this for frontend)
- Run Prisma migration checks in CI

---

9. Frontend Architecture Concerns (Severity: LOW-MEDIUM)

Issue	Detail
Dual data access	Frontend reads directly from Supabase AND calls the backend API — inconsistent data flow
Large page components	Dashboard pages are 10K-50K lines each, not decomposed
No real-time updates	Dashboard shows static data; no WebSocket/SSE for live call status
No error boundaries	No graceful degradation on component failures

Recommendation: Route all data access through the API for consistent authorization and logging. Break down large page components. Add real-time call status updates via WebSocket.

---

10. Database Improvements Needed (Severity: LOW-MEDIUM)

Issue	Recommendation
Prisma defined but not used at runtime	Either commit to Prisma or remove it; current dual-stack is confusing
No customer de-duplication	Add unique constraint on `(business_id, phone)`
No soft deletes	Add `deleted_at` column for recoverable deletions
No audit trail	Add `audit_log` table for sensitive operations

Issue	Recommendation
TEXT primary keys	Consider migrating to native UUID type for better index performance
N+1 queries in campaign scheduler	Batch queries for customer eligibility checks

Recommended Target Architecture

Phased Implementation Roadmap

Phase 1 — Make It Sellable (4-6 weeks)

Goal: Accept payments and onboard paying customers.

Task	Priority	Effort
Stripe integration (subscriptions + checkout)	P0	1 week
Tier-based call limits middleware	P0	3 days
Usage metering (calls, minutes)	P0	3 days
Per-business phone number provisioning	P0	1 week
Telnyx webhook signature verification	P0	1 day
Customer-facing billing portal	P1	3 days
Basic monitoring (Sentry)	P1	1 day
Env var validation at startup	P1	1 day

Phase 2 — Make It Scalable (4-6 weeks)

Goal: Handle 100+ businesses reliably.

Task	Priority	Effort
Decompose monolith into modules	P0	2 weeks
BullMQ job queue for schedulers	P0	1 week
Separate worker processes	P0	3 days
CI/CD pipeline (GitHub Actions)	P1	2 days
Integration & E2E tests	P1	1 week
Structured JSON logging	P1	2 days

Task	Priority	Effort
Request validation with Zod	P1	3 days
API versioning (`/api/v1/`)	P2	1 day

Phase 3 — Make It Enterprise-Ready (6-8 weeks)

Goal: Land larger accounts and reduce churn.

Task	Priority	Effort
Team/role management	P1	1 week
Usage dashboards for businesses	P1	1 week
Real-time call status (WebSocket)	P1	3 days
DNC list management (TCPA compliance)	P0	3 days
Customer opt-out tracking	P0	2 days
Audit trail logging	P1	3 days
API documentation (OpenAPI)	P2	3 days
Number porting support	P2	1 week
Custom voice training	P2	2 weeks
White-label / custom branding	P3	2 weeks

Compliance Considerations

For a calling SaaS targeting US/Canadian SMBs, you must address:

- **TCPA (Telephone Consumer Protection Act)**: Requires prior express consent for automated calls. You need consent tracking per customer.
- **DNC (Do Not Call) Registry**: Must check the National DNC Registry before outbound calls. Implement a `b2b_dnc_list` table and pre-dial checks.
- **CRTC (Canada)**: Similar rules for Canadian numbers. Requires Canadian DNC list checking.
- **Call recording disclosure**: If calls are recorded/transcribed, disclosure must be made at the start of the call (check per-state laws for one-party vs. two-party consent).
- **Caller ID rules**: FCC requires accurate caller ID. Per-business numbers solve this.
- **STIR/SHAKEN**: Telnyx handles this at the trunk level, but ensure your numbers are properly attested.

This review was prepared on February 8, 2026, based on analysis of the nemo-b2b-api, nemo-b2b-livekit, and nemo-b2b-web codebases.