Technical Appendix

Supporting detail for: Rebuilding Broken CRM Data for a Scaling Sales Agency • Client name withheld

1. Data Model — Separating Permanent Records From Active Campaigns

The core architectural decision was decoupling permanent identity records (a company, a person) from temporary, campaign-specific data (a given outreach effort for a given client). Without this separation, the same person being targeted by two different clients creates crossed data and cross-client pollution.

```
Account (Company) → Person (Contact) → Prospect → Opportunity → Project
```

> **Account & Person** hold permanent identity data — canonical domain, verified email, mobile number, a global do-not-contact flag — and exist once in the system regardless of how many clients are interested in them.
>
> **Prospect** represents one specific client's active pursuit of that person. Stage, outreach notes, and call outcomes live here, scoped to that one campaign.
>
> **Project** is the contracted delivery container per client. A closed-won deal automatically provisions a new delivery project via workflow automation.

Practical effect: if two different agency clients are both targeting the same person at the same company, that person's record stays clean and singular, while two independent Prospect records track each client's outreach with zero overlap between them.

2. Multi-Tenant Access — Why Native Teams Were Rejected

Two options were evaluated for segregating rep and project-manager access across clients:

Option	Verdict
HubSpot Native Teams	Free up to 10 teams, but costs scale sharply beyond that (roughly $200/month per additional 100 teams), and doesn't cleanly support a project manager overseeing several client projects at once.
Custom Project Assignment model **(chosen)**	A custom model linking rep logins directly to specific projects. Avoids the licensing penalty entirely as the agency scales from 7 toward 40+ client projects, and natively supports one PM managing multiple projects.

3. One Canonical Import Template, Not Custom Parsers Per Client

Rather than writing bespoke import logic to handle each incoming client spreadsheet's own layout, the burden was reversed: a single canonical Excel template was built, with locked dropdowns and pre-validated schemas. Any client workbook has to conform to this template before it's allowed into the ingestion pipeline. This traded early flexibility for consistency at scale — see the case study's 'What I'd Do Differently' for why this should have been enforced from day one rather than introduced after the first round of custom parsing.

4. Activity Tracking, Not Flat Spreadsheet Fields

Legacy fields like a free-text "Next Follow-up Date" or "Next Steps" column were explicitly excluded from the rep-facing CRM view. Reps now work from native CRM tasks and calendar-synced meeting records instead, which prevents double data entry and makes activity reporting possible in the first place — flat text fields can't be reported on.

5. A Real Integration Gotcha Worth Recording

HubSpot's versioned Lead API requires internal pipeline enum keys (e.g. `IN_PROGRESS`, `OPEN_DEAL`) rather than the friendly labels shown in the UI ("In Progress", "MQL"). Passing the display label instead of the enum key fails silently — no error, the record just doesn't update. This cost real debugging time early on and needed its own transformation layer to catch before it reached the API.

This appendix supports the case study "Rebuilding Broken CRM Data for a Scaling Sales Agency." Figures referenced here are recorded as measured during the Phase 1 build, not projected.