1. Executive Summary

VibeCRM is an **Agent-Native CRM Builder**. It allows users to "vibe" (prompt) a full-stack, relational CRM into existence. Unlike generic app builders, VibeCRM uses **Refine.dev** as a standardized chassis to ensure 100% reliability and **Context Graphs** to capture the reasoning behind business data.

2. Technical Stack (The Chassis)

- **Frontend:** Next.js (App Router), TypeScript, **Tailwind CSS**, and **shadcn/ui**.
- **Logic Engine: Refine.dev** (Headless mode).
- **Database: Supabase** (Postgres) with Row-Level Security (RLS).
- **AI Orchestration:** Claude 3.5 Sonnet via Anthropic API.

---

3. Core Features (MVP Scope)

3.1 The "Vibe" Generator

- **Input:** A single, detailed text area for the user's business description.
- **The Architect Agent:** A specialized Claude prompt that outputs a **JSON CRM Spec** (Tables, Columns, Relationships).
- **The Provisioner:** A backend service that creates isolated schemas in Supabase based on the JSON Spec.

3.2 Dynamic Refine Renderer

- **Resource Mapping:** A system that takes the JSON Spec and dynamically generates Refine `<Resource />` components.
- **shadcn UI Mapping:**
  - **Lists:** `useTable` (Refine) → `DataTable` (shadcn).
  - **Forms:** `useForm` (Refine) → `AutoForm` (shadcn primitives).
  - **Navigation:** A dynamic sidebar built from the resource map.

3.3 The Context Graph (System of Reasoning)

- **Decision Traces:** Every record creation or update must trigger a `decision_trace` entry.
- **Trace Schema:**
  - `record_id`: UUID

    - `intent`: The user's prompt or the AI's logic.
    - `precedent`: Why this change was made (e.g., "Applied discount based on holiday policy").

---

4. System Architecture & Data Flow

1. **Intent Layer:** User submits prompt.
2. **Logic Layer:** Claude parses intent → Validates against CRM patterns → Outputs JSON.
3. **Persistence Layer:** Backend executes SQL in Supabase → Stores "CRM Metadata" in a `projects` table.
4. **UI Layer:** Frontend fetches `projects` metadata → Hydrates Refine Resources → Renders shadcn components.

---

5. Implementation Guidelines (Strict)

- **Rule 1: No Hardcoding.** The UI must be entirely driven by the JSON Spec. If a user adds a "Property" table, the UI must appear without a code deployment.
- **Rule 2: Trace Everything.** No data should change without an accompanying entry in the `context_graph` table.
- **Rule 3: Type Safety.** Every generated piece of code must be validated via `npm run type-check`.
- **Rule 4: Headless Refine.** Use `@refinedev/core` for logic and `@refinedev/react-hook-form` for state. Do not use Refine's pre-styled Ant Design components; use **shadcn/ui**.

---

6. Definition of Done (MVP)

- User can type: "Build a CRM for a fitness coach to track clients and meal plans."
- System generates a working dashboard with "Clients" and "Meal Plans" tabs.
- User can add a client, and the system records the **Intent** (e.g., "Onboarding new high-protein client").
- The entire app passes a full TypeScript check.

---

7. Next Steps for AI Agent

"Using this PRD, start by defining the **TypeScript Interface** for the **CRM Specification JSON**. This interface must account for table structures, relational foreign keys, and UI hints (icons, labels)."