Build a complete, single-user internal CRM web application for my IT Services and Development Agency, **Devs On Steroids** (website: [www.devsonsteroids.com](http://www.devsonsteroids.com)). This is a personal management tool for me (TroubleShooter, Co-Founder & CEO) to track clients, projects, invoices, team assignments, salaries, internal requests, and budgets for our global client base. We specialize in **custom-coded solutions** (from-scratch development, no off-the-shelf platforms like Shopify or WordPress). No multi-user roles or client-facing interfaces—only I will use it with full admin access via a single login (email: [insert your email], password hashed). Use **React.js** (with **Vite** for fast setup) for the frontend, styled with **Tailwind CSS** for a responsive, mobile-first UI (bundle <5MB). Use **Supabase** for the backend: **PostgreSQL** for the database, **Supabase Auth** (JWT-based) for secure login, **Supabase Storage** for file uploads, **Supabase Realtime** for live updates, and **Supabase Edge Functions** for email notifications. Deploy the entire app (frontend, Edge Functions, and any custom Node.js components) to **Railway.app** (use Railway's Node.js support for backend parts and static hosting for frontend; integrate Supabase as the cloud backend). Use **shadcn/ui** or **Headless UI** for polished forms/tables, with dark/light mode toggle. Include comprehensive error handling, **Zod** validation, and a demo setup with seed data.

### **Company Context**

Devs On Steroids provides global, **custom-coded IT services**: from-scratch website development (brand-based, corporate), custom e-commerce platforms (carts, subscriptions, multi-vendor), PWAs (offline-first, push notifications), mobile apps (React Native/Flutter, iOS/Android), marketplaces (vendor dashboards, commissions), UI/UX design, graphics/video/illustrations (GVI), APIs, SEO, and maintenance—all built with custom code (e.g., Node.js/React backends, no Shopify/WordPress). Clients are worldwide (e.g., **Empty Advertising Inc.** from Canada, others in US/EU). I need to track client interactions, past/current/upcoming projects, invoices, team assignments, employee salaries, internal team requests, and development team budgets to stay organized as CEO.

### **Employee List (For Assignment, Salary, and Budget Tracking)**

Use this fixed list for dropdown assignments in the UI—no user management or role-based restrictions. Any employee can be assigned any project/task, regardless of department/role. Departments are for filtering only (e.g., to group Development tasks). Link to salary/budget sections:

| **Employee Name** | **Role** | **Department** |
| --- | --- | --- |
| TJ | Co-Founder | Leadership |
| Fusion | Senior FrontEnd | Development |
| AlgoX | Senior BackEnd | Development |
| Zenx | FrontEnd | Development |
| White Bear | FrontEnd | Development |
| Flux | Head of Mobile App | Mobile |
| Codx | Head of Development | Development |
| Byte | Head of Creative | Creative |
| Red | Head of UI/UX | UI/UX |
| Design Artistic | Head of GVI | Graphics/Video |

### **UI/UX Structure**

* **Dashboard Home**: Sidebar navigation (Clients, Projects, Invoices, Features Catalog, Reports, Salaries, Team Requests, Budgets, Settings). Top bar: Global search (across all sections), profile (logout, theme toggle). Hero section: Stat cards (e.g., Active Clients: 25, Pending Payments: $15K, Upcoming Projects: 8, Overdue Invoices: 3, Total Salary Outlay: $50K, Budget Variance: +5%). Mini-charts via **Recharts** (e.g., revenue trend line, project type pie, monthly budget bar).
* **Tables**: Use **TanStack Table** for sortable, filterable lists (e.g., columns: Employee Name, Status, Amount, Due Date). Features: Pagination (10-50 rows), filters (department, type, status, date range), export to CSV/JSON.
* **Forms**: Modal-based (e.g., Add Client/Project/Salary Entry) with **Zod** validation, auto-save drafts, rich text (**React-Quill**) for descriptions/notes.
* **Views**:
  + Calendar view (**react-big-calendar**) for project deadlines, milestones, upcoming starts, salary due dates, request due dates, and budget cycles.
  + Kanban board (**react-beautiful-dnd**) for Upcoming Queue and Team Requests (drag-drop prioritization).
* **Notifications**: In-app toasts (**react-hot-toast**) for actions (e.g., “Salary for Fusion logged”). Email alerts via **Supabase Edge Functions** (e.g., Postmark integration) for deadlines/overdue items (e.g., “Project X due tomorrow”, “Salary payment due”, “Team Request for Trae Subscription expires in 3 days”).
* **Files**: Upload to **Supabase Storage** (max 10MB per file, e.g., contracts, designs, salary receipts, budget docs). Store public URLs for access.
* **Error Handling**: Client-side (toast for invalid inputs) and server-side (400/422 for API errors, e.g., duplicate salary entry). Alerts for budget overages (e.g., red highlight if expenses > budget).

### **Detailed Features**

1. **Client Management**
   * **Route**: /clients/:id (e.g., /clients/empty-advertising-inc). Tabs: Overview, Past Projects, Current Projects, Upcoming Queue, Invoices/Payments, Notes.
   * **Overview Tab**: Editable form fields:

| **Field** | **Type** | **Example** |
| --- | --- | --- |
| Name | String | Empty Advertising Inc. |
| Company | String | Empty Advertising |
| Location | JSON | { country: "Canada", city: "Toronto", timezone: "America/Toronto" } |
| Contacts | JSON | [{ email: "contact@emptyadv.com", phone: "+1-416-555-1234" }, …] |
| Website | String | <https://emptyadv.com> |
| Total Spend | Number | $50,000 (auto-calculated from invoices) |
| Notes | Rich Text | “Signed custom e-comm contract Jan 2023; prefers quick turnarounds” |
| Status | Enum | Active/Inactive/Lead |

* + **Past Projects Tab**: Timeline (vertical cards) or table. For each project:

| **Field** | **Description** | **Example/Data Type** |
| --- | --- | --- |
| Name | Title | “Custom E-commerce v2” (string) |
| Type | Multi-select | [Website, E-comm, PWA] (array) |
| Start Date | Actual start | 2023-01-15 (date) |
| End Date | Actual end | 2023-06-30 (date) |
| Description | Rich text | “Custom-coded React/Node platform with RBAC, i18n” (string) |
| Quotation | Breakdown + PDF | Features selected, total $25K (JSON, downloadable PDF) |
| Actual Scope | Rich text + files | “Added 3 custom APIs, scope creep noted” (string + Supabase Storage URLs) |
| Deliverables | List + files | “Live URL, code zip, handover doc” (array + URLs) |
| Assigned Employees | Multi-select | “Fusion, AlgoX” (array, any employee) |
| Hours | Per employee | [{ employee: “Fusion”, hours: 200 }, …] (array) |
| Status History | Auto/manual log | [{ status: “Quoted”, date: “2023-01-10” }, …] (array) |
| Cost Breakdown | Quoted vs. Actual | { quoted: $25K, actual: $28K } (JSON) |
| Lessons Learned | Textarea | “Client delayed feedback” (string) |

* + **Current Projects Tab**: Table (filter by dept: Development/Mobile/UIUX/Creative/GVI). Columns: Name, % Complete (progress bar), Assignees (any employee), Bottlenecks (textarea, e.g., “Waiting client assets”), Next Milestone (date). Real-time updates via **Supabase Realtime**.
  + **Upcoming Queue Tab**: Kanban board with columns (Backlog, Ready to Quote, Quoted, Scheduled). Cards include:
    - Name, Client Link, Probable Start (range: “Nov 1-7, 2025”), Priority (High/Med/Low), Estimated Duration (weeks), Pre-requisites (checkboxes: “Brief received?”, “Budget approved?”), Potential Revenue ($ range, manual input).
    - Drag-drop to reorder; auto-sort by priority or start date.
    - Calendar sync: Probable start dates appear in react-big-calendar.
  + **Invoices/Payments Tab**: See below.
  + **Bulk Actions**: Add clients via form or CSV import; archive inactive clients (soft delete with is\_active flag).
  + **Edge Cases**: Prevent duplicate clients (check name+company), warn on incomplete forms, handle failed file uploads.

1. **Project Management**
   * **Global Projects Page**: /projects table aggregating all clients. Filters: Status (Past/Current/Upcoming), Dept, Type, Date Range.
   * **Create/Edit Project**: Modal form (from client page or global). Auto-link to client. Fields mirror Past Projects table, plus:
     + **Milestones**: Sub-tasks array (e.g., [{ name: “Custom Design”, due: “2025-02-01”, progress: 50%, assignee: “Red” }, …]).
     + **Risks/Changes**: Log array (e.g., [{ date: “2025-01-10”, note: “Added custom referral system +$2K” }]).
     + **Hours Tracking**: Form to log hours per employee (dropdown + number input, auto-total).
   * **Queue Integration**: New projects auto-add to Upcoming Queue (Backlog column). “Start Project” button moves to Current Projects, sets start date.
   * **Calendar View**: Show milestones, deadlines, and probable starts in react-big-calendar, with click-to-view details.
2. **Features Catalog (For Quoting)**
   * **Dedicated Page**: /features – Editable table of reusable features/services. Categories: Development, Design, Mobile, E-comm, Other. No default prices—you set prices manually per project/quote.

| **Feature Name** | **Category** | **Description** | **Notes** |
| --- | --- | --- | --- |
| Multilingual Support (i18n) | Development | Full locale switching, RTL | Per language cost varies |
| Role Based Access Control (RBAC) | Development | Admin/Employee panels | Custom roles possible |
| Content Management System (CMS) | Development | Custom-coded backend | From-scratch, no Strapi |
| Referral Management System | E-comm | Affiliate tracking | Custom Stripe payouts |
| Custom UI/UX Design | Design | Wireframes to prototypes | Figma exports |
| Payment Gateway Integration | E-comm | Custom Stripe/PayPal | Subscriptions support |
| SEO Optimization | Other | On-page, schema markup | Analytics setup |
| API Development/Integration | Development | Custom REST/GraphQL | Auth + rate limiting |
| PWA Features | Development | Offline cache, push notifications | Service worker |
| Cross-Platform Mobile App | Mobile | Custom React Native | iOS/Android |
| Custom Illustrations | GVI | Vector graphics (10 assets) | Source files |
| Video Editing/Animation | GVI | 2-min promo clip | 4K export |
| Inventory Management | E-comm | Stock alerts, multi-warehouse | Custom ERP sync |
| Subscription Billing | E-comm | Recurring + trials | Churn analytics |
| Marketplace Vendor Dashboard | Marketplace | Listings, payouts | Custom commission calc |
| Performance Optimization | Other | Speed <2s, Lighthouse 95+ | Custom bundle analysis |
| Security Audit | Other | OWASP top 10 fixes | Report + cert |
| Maintenance Retainer | Other | Monthly updates/support | Hours-based |
| Custom Dashboard Analytics | Development | Charts/reports | Recharts-based |
| Brand Identity Kit | Creative | Logo, colors, guidelines | Full package |

* + **Usage**: In Project/Quote form, multi-select features from catalog. Manually input price per feature (e.g., “i18n: $600”). Auto-calculate total with tax/discount sliders (e.g., 10% off for repeat clients). Save as JSON breakdown.
  + **Quote Generation**: Quotations must be detailed PDFs matching the example invoice format (but labeled as "Quotation" instead of "Invoice"). Use **jsPDF** or **pdf-lib** for client-side PDF creation. Layout details:
    - **Header**: Company logo (uploadable in Settings) on left, "Quotation" title on right, in a blue banner.
    - **Bill To Section**: "Bill To:" on left, customer name, email, website below it.
    - **Quote Info**: Quote # (auto-increment, e.g., DOS-Q001), Date of Issue, Due Date.
    - **Item Table**: Columns: ITEM/SERVICE (e.g., UI/UX), DESCRIPTION (detailed, e.g., "Driver + Client + Admin"), QTY/HRS (e.g., 3), RATE (e.g., $100), AMOUNT (e.g., $300). Rows for each selected feature with manual prices.
    - **Totals**: Right-aligned box with Subtotal, Payment (if partial), Tax Rate (configurable), Tax, Balance.
    - **Footer**: "Quote Placed By" with name (e.g., M. A. ROMAN, CO FOUNDER), email, phone (+1 929 346 1357 USA, +880 16280 78937 BD), website. Include a horizontal line separator.
    - Allow emailing the PDF via Supabase Edge Function. Ensure professional, clean formatting with borders, fonts (Arial/Sans-serif), and alignment as in the example.

1. **Invoice and Payment Management**
   * **Global Invoices Page**: /invoices table: Number (auto: DOS-001), Client, Amount, Status (Draft/Sent/Paid/Partial/Overdue), Due Date. Filters: Paid/Pending/Overdue.
   * **Per-Client Tab**: List invoices with totals (e.g., Pending: $5K, Billed: $100K).
   * **Generate Invoice**: From project quote or manual entry. Fields:
     + Number (auto-increment: DOS-001, DOS-002, …).
     + Client (linked), Items (features + manual prices), Subtotal, Tax (configurable: 0-18%, e.g., 5% GST Canada), Total, Due Date, Terms (Net 30).
     + Template: Match the detailed example layout exactly, labeled as "Invoice".
       - **Header**: Company logo on left, "Invoice" title on right, in a blue banner.
       - **Bill To Section**: "Bill To:" on left, customer name (e.g., Empty Advertising), email (e.g., beckmajdell11@gmail.com), website (e.g., [www.emptyad.com](http://www.emptyad.com)).
       - **Invoice Info**: Invoice # (e.g., 5592), Date of Issue (e.g., July 2, 2025), Due Date (e.g., July 2, 2025).
       - **Item Table**: Columns: ITEM/SERVICE (e.g., UI/UX), DESCRIPTION (detailed, e.g., "Driver + Client + Admin"), QTY/HRS (e.g., 3), RATE (e.g., $100), AMOUNT (e.g., $300). Rows for each item.
       - **Totals**: Right-aligned box with Subtotal (e.g., $1,900.00), Payment (e.g., $950.00), Tax Rate, Tax (e.g., $0.00), Balance (e.g., $950.00).
       - **Footer**: "Invoice Placed By" with name (e.g., M. A. ROMAN (CO FOUNDER, DEVS ON STEROIDS)), email (hello@devsonsteroids.com, devsonsteroids@gmail.com), phone (+1 929 346 1357 (USA), +880 16280 78937 (BD)), website ([www.devsonsteroids.com](http://www.devsonsteroids.com)). Include a horizontal line separator.
     + Use **jsPDF** to generate downloadable PDFs with this exact formatting (borders, fonts, alignment).
   * **PDF Download**: Button to download; email via **Supabase Edge Function** (Postmark/SendGrid).
   * **Payment Tracking**: Log payments (full/partial) with fields: Amount, Date, Method (Bank/Wire/Stripe). Auto-update invoice status (e.g., $5K partial → Partial).
   * **Reminders**: Manual “Resend” button; auto-emails for overdue invoices (weekly cron via Edge Function).
   * **Edge Cases**: Prevent duplicate invoice numbers; warn if total mismatches items.
2. **Reporting and Analytics**
   * **Reports Page**: Tabs: Overview, Client-Wise, Project-Type Revenue, Dept Load, Payment Aging, Employee Performance, Salary Summary, Budget Overview.
     + **Overview**: Total revenue, active projects, overdue invoices, client count, total salary outlay.
     + **Client-Wise**: Spend per client, project count, avg project duration.
     + **Project-Type Revenue**: Breakdown by type (e.g., Custom E-comm: $200K, 40%).
     + **Dept Load**: Hours assigned vs. completed per dept (e.g., Dev: 80% capacity).
     + **Payment Aging**: Invoices >30/60/90 days overdue.
     + **Employee Performance**: Hours logged, projects completed, avg time per task.
     + **Salary Summary**: Monthly/annual totals, unpaid salaries.
     + **Budget Overview**: Variance per category (e.g., Training: $2K over budget).
     + **Custom Query**: Filterable table (e.g., “Projects >$10K in 2024, Dev dept”).
   * **Charts**: Use **Recharts** for visuals (e.g., bar for revenue by type, line for monthly income, pie for budget categories). Export to PNG/PDF.
   * **Export**: All reports downloadable as PDF (**html-to-pdfmake**) or Excel (**xlsx**).
3. **Salaries Management**
   * **Dedicated Page**: /salaries – Table of employees with monthly salary settings and payment history. Link to employee list.
     + **Set Salaries**: Form to set/update monthly salary per employee (e.g., Fusion: $5K/month, effective from Jan 2025). Auto-calculate annual ($60K).
     + **Payment History**: Table per employee/month:

| **Field** | **Description** | **Example/Data Type** |
| --- | --- | --- |
| Month/Year | Period | Jan 2025 (date) |
| Base Salary | Set amount | $5,000 (numeric) |
| Bonus | Optional | $500 (numeric) |
| Total Due | Auto-calc | $5,500 (numeric) |
| Paid Amount | Logged | $5,500 (numeric) |
| Payment Date | Actual | 2025-01-31 (date) |
| Method | Dropdown | Bank Transfer, PayPal |
| Receipt | File Upload | Supabase Storage URL |
| Status | Enum | Paid/Unpaid/Partial |

* + - **History View**: Timeline or table for past payments (e.g., filter by employee/month). Totals (e.g., YTD paid: $30K).
    - **Calendar Integration**: Due dates (end-of-month) in react-big-calendar; auto-notify 3 days before (e.g., “Salary for AlgoX due”).
    - **Bulk Actions**: Log multiple payments via CSV; mark as paid with receipt upload.
    - **Edge Cases**: Prevent duplicate payments; warn if unpaid >30 days.

1. **Team Requests**
   * **Dedicated Page**: /team-requests – Table or Kanban for internal requests from the development team (e.g., tool needs, subscriptions).
     + **Log Request**: Form (e.g., Title: “Trae Subscription needed for Project X”, Description: “Renew for 6 months”, Requester: Employee dropdown, Category: Tools/Training/Hardware, Estimated Cost: $100, Urgency: High).
     + **Assign Task/Date**: Set due date (e.g., “Approve by Nov 15, 2025”), assignee (any employee), status (Pending/Approved/Denied/In Progress).
     + **Kanban Columns**: New, In Review, Assigned, Completed.
     + **Notifications**: Auto-emails 7/3 days ahead of due date (via Edge Function, e.g., “Trae Subscription request expires soon”). In-app toasts on status changes.
     + **History**: Archive completed requests; link to budgets for cost tracking.
     + **Calendar Integration**: Due dates in react-big-calendar.
     + **Edge Cases**: Prevent duplicate requests (check title+requester); auto-close after 90 days if inactive.
2. **Budgets Management (Focused on Development Team)**
   * **Dedicated Page**: /budgets – Monthly/quarterly budget tracker for development team (e.g., salaries, training, tools).
     + **Set Budget**: Form for new period (e.g., Month: Jan 2025). Categories with allocations:

| **Category** | **Description** | **Allocated Amount** | **Example** |
| --- | --- | --- | --- |
| Salaries | Employee pay | $20,000 | Auto-pull from salaries |
| Training Costs | Courses/conferences | $2,000 | e.g., React Native workshop |
| Tools/Subscriptions | Software (e.g., Trae, Figma) | $1,500 | e.g., $100 Trae renew |
| Hardware/Equipment | Laptops, monitors | $3,000 | Receipts uploaded |
| Other (Travel, Misc) | Flexible | $500 | Custom entries |

* + - **Expense Tracking**: Log expenses (linked to requests/salaries): Amount, Date, Category, Description, Receipt (file), Status (Approved/Paid).
    - **Variance Report**: Auto-calc (Actual vs. Budgeted, e.g., +$500 over in Tools). Red alerts for overages (>10%).
    - **Totals**: Monthly summary (e.g., Total Budget: $27K, Spent: $25K, Remaining: $2K). Annual roll-up.
    - **Filters**: By month/category/employee.
    - **Calendar Integration**: Budget cycle starts (e.g., 1st of month) in react-big-calendar.
    - **Edge Cases**: Warn on over-budget; prevent negative remaining; auto-rollover unused to next month.

1. **Settings**
   * **Company**: Upload logo (Supabase Storage), set address, multi-currency (USD default, CAD/EUR via **openexchangerates** API).
   * **Taxes**: Per-country rates (e.g., Canada GST 5%).
   * **Emails**: Configure SMTP for Edge Functions (e.g., Postmark API key).
   * **Backups**: Manual DB export button; auto-backup to Supabase Storage (weekly).

### **Supabase Integration**

* **Database Schema (PostgreSQL Tables)** – Add new tables for sections:
  + **clients**: { id (uuid, PK), name (text), company (text), location (jsonb), contacts (jsonb), website (text), total\_spend (numeric), notes (text), is\_active (boolean), created\_at (timestamp), updated\_at (timestamp) }
  + **projects**: { id (uuid, PK), client\_id (uuid, FK), name (text), type (text[]), start\_date (date), end\_date (date), description (text), quotation (jsonb), actual\_scope (text), deliverables (jsonb), assignments (jsonb), milestones (jsonb), status\_history (jsonb), cost\_breakdown (jsonb), lessons\_learned (text), created\_at, updated\_at }
  + **upcoming\_queue**: { id (uuid, PK), project\_id (uuid, FK), stage (enum), probable\_start (jsonb), priority (enum), duration\_weeks (int), prereqs (jsonb), potential\_revenue (numeric), created\_at, updated\_at }
  + **invoices**: { id (uuid, PK), project\_id (uuid, FK), client\_id (uuid, FK), number (text, unique), items (jsonb), subtotal (numeric), tax (numeric), total (numeric), due\_date (date), status (enum), payments (jsonb), created\_at, updated\_at }
  + **features**: { id (uuid, PK), name (text), category (enum), description (text), notes (text), created\_at, updated\_at }
  + **employees**: { id (uuid, PK), name (text), role (text), department (enum), created\_at }
  + **salaries**: { id (uuid, PK), employee\_id (uuid, FK), month\_year (date), base\_salary (numeric), bonus (numeric), total\_due (numeric), paid\_amount (numeric), payment\_date (date), method (enum), receipt\_url (text), status (enum), created\_at, updated\_at }
  + **team\_requests**: { id (uuid, PK), title (text), description (text), requester\_id (uuid, FK), category (enum: tools/training/hardware/other), estimated\_cost (numeric), urgency (enum), due\_date (date), assignee\_id (uuid, FK), status (enum: pending/review/assigned/completed), created\_at, updated\_at }
  + **budgets**: { id (uuid, PK), period\_month (date), category (enum), allocated\_amount (numeric), expenses (jsonb: [{amount, date, desc, receipt\_url}]), actual\_total (numeric), variance (numeric), created\_at, updated\_at }
* **Supabase Auth**: Single user (your email). JWT-based, with password reset via Supabase Auth UI.
* **Supabase Storage**: Bucket crm-files for uploads (e.g., receipts, requests docs).
* **Supabase Realtime**: Subscribe to projects, invoices, salaries, team\_requests, budgets for live updates.
* **Edge Functions**: Email notifications (e.g., salary dues, request reminders, budget alerts) using Postmark/SendGrid. Cron for weekly summaries.
* **API Queries**: Use Supabase JS client for CRUD (e.g., supabase.from('salaries').select('\*')). Paginate large tables (limit: 50).

### **Technical Specs**

* **Frontend**: React (Vite), Tailwind CSS, shadcn/ui, Recharts, react-big-calendar, react-beautiful-dnd, React-Quill, react-hot-toast.
* **Backend**: Supabase (PostgreSQL, Auth, Storage, Realtime, Edge Functions).
* **APIs**: REST-like via Supabase queries. Validate inputs with Zod. Handle errors (400/422 with messages).
* **Integrations**: Supabase for all backend; **openexchangerates** for currency; **Postmark** for emails.
* **Security**: HTTPS, Supabase row-level security (RLS: only your auth user can access), rate-limit API calls.
* **Performance**: Index key fields (e.g., employee\_id, period\_month). Lazy-load tables, cache currency rates.
* **Edge Cases**:
  + Duplicate entries: Check uniqueness (e.g., salary per month/employee).
  + File uploads: Retry on failure, limit to 10MB.
  + Currency: Store in USD, display converted (CAD/EUR).
  + Offline: Cache recent data in localStorage, sync on reconnect.
  + Budget Overages: Auto-flag and notify if >10% overrun.
* **Testing**: **Vitest** for frontend (80% coverage: CRUD, forms). Supabase SQL tests for queries.
* **Deployment**: Railway.app (frontend: static/Node build; Edge Functions: Node.js service). Provide .env template (SUPABASE\_URL, SUPABASE\_KEY, POSTMARK\_KEY, RAILWAY\_ENV). Steps: 1) Connect Supabase project, 2) Push code to Railway Git, 3) Set env vars, 4) Deploy services.

### **Deliverables**

* **Source Code**: Zipped repo (/client, /server/edge-functions, /docs). Include Supabase migration scripts (SQL).
* **Demo Setup**: Seed DB with 10 fake clients (e.g., Empty Advertising Inc.), 20 projects, 15 invoices, 12 salary entries (6 months, mixed statuses), 5 team requests (e.g., "Trae Subscription"), 3 budgets (Jan-Mar 2025 with expenses).
* **Docs**: README with setup (npm install, env vars, Supabase config, Railway deploy). 5-min video script for local setup/demo.
* **Extensibility**: Hooks for future AI (e.g., OpenAI API for budget forecasts or request summaries).

If unclear, prioritize an intuitive, CEO-friendly CRM emphasizing custom coding tracking and internal ops. Generate a polished, bug-free app ready for immediate use on Railway.