ChatGPT

Blueprint Arsitektur End-to-End (Target-State)

UGC BCP adalah **single source of truth** untuk modul CRM, Ticketing, DSO/AR, KPI Engine, dan Dashboard. Masing-masing modul memiliki **batasan domain** dan **kepemilikan data** yang jelas, namun terintegrasi dengan lancar. Sebagai contoh, modul CRM bertanggung jawab penuh atas **data pelanggan** (leads, kontak, akun, peluang, aktivitas), sedangkan modul Ticketing menangani **permintaan internal** (tiket operasional, kuotasi, respon). Modul DSO/AR ("Days Sales Outstanding / Accounts Receivable") mengelola **piutang** berdasarkan data penjualan (yang dihasilkan dari CRM) dan pembayaran, sedangkan KPI Engine menghitung metrik bisnis (KPI marketing/sales, SLA, response time) menggunakan data CRM, Ticketing, dan AR. Dashboard mengagregasi ringkasan data dari semua modul ini. Integrasi antar-modul diatur melalui **kunci referensi bersama** (misalnya ID akun atau pelanggan). Setiap modul wajib mengikuti RLS (Row-Level Security) dan hanya mengakses data yang diizinkan sesuai peran (role) — **tepat 15 peran** seperti tertera di BLUEPRINT (Director, super_admin, Marketing Manager, Marcomm, DGO, MACX, VSDO, sales manager, salesperson, sales support, EXIM Ops, domestics Ops, Import DTD Ops, traffic & warehous, finance). Tidak boleh ada peran tambahan atau dikurangi.

Desain Skema Database (Rancangan Tabel)

Skema database dirancang sejak nol dengan tabel terpisah per modul, relasi jelas, dan audit log terpusat. Contoh entitas utama:

- **Tabel Master**: `users` (dengan kolom `id`, `role`, `department`, RLS aktif); `roles`, `departments`.
- **CRM**: `leads` (id, nama, email, sumber, *triage_status* enum {New, In Review, Qualified}, assigned_user), `opportunities` (id, lead_id, account_id, *stage* enum {Prospecting, Negotiation, Won, Lost, Dismissed}, value, next_action_date), `accounts` (id, company_name, industry, owner_user_id), `contacts` (id, account_id, name, email), `activities` (id, account_id/lead_id/ opportunity_id, type, due_date, status {Planned, Completed}, owner_user_id). Terdapat juga tabel bantu `lead_handover_pool` untuk pencatatan pengalihan lead, dan `activity_history` untuk log aktivitas. Setiap perubahan penting dicatat di `audit_logs` (tabel tersendiri) dengan kolom `timestamp`, `user_id`, `table_name`, `record_id`, `operation`, `changes`.
- **Ticketing**: `tickets` (id, title, description, status enum {Open, Need Response, In Progress, Waiting Customer, Need Adjustment, Pending, Resolved, Closed}, priority enum, type enum, created_by, assigned_to, created_at, updated_at, close_outcome enum {won,lost}, close_reason, competitor_name, competitor_cost). Relasi: `ticket_comments` (id, ticket_id, author_id, content), `ticket_attachments` (id, ticket_id, url), `ticket_assignments` (id, ticket_id, user_id, assigned_at). Indeks dibuat pada kolom yang sering difilter (misal status, assigned_to). RLS dipastikan aktif dan kebijakan mengikat tiket hanya dapat diakses oleh departemen terkait.
- **DSO/AR**: `invoices` (id, account_id, date_issued, due_date, amount, status enum {Outstanding, Paid, Overdue}), `payments` (id, invoice_id, date_paid, amount), `customers` (sinkronisasi dengan `accounts` CRM), serta tabel `ar_aging` atau view untuk menghitung DSO. Master `currencies`, `tax_rates` juga diperlukan bila ada.

- **KPI Engine**: Tabel `kpi_config` (id, name, formula, target), `kpi_results` (id, kpi_config_id, period, value). Data KPI dihitung via SQL/RPC rutin (tidak disimpan ganda di modul lain) sehingga data penjualan (dari CRM) dan tiket (dari Ticketing) diolah untuk menghasilkan metrik akhir.
- **Log & Audit**: Tabel `audit_logs` (id, user_id, module, action, details, timestamp). Ini untuk **no double entry** dan compliance: setiap modul menulis log di sini untuk transaksi penting (crucial state changes, error).
- **Supabase Storage Buckets**: Untuk upload file (lampiran tiket, dokumen pendukung).

Semua FK dan constraint dikunci ketat. Misalnya, `invoices.account_id` adalah FK ke `accounts.id`. Enum diatur dengan CHECK constraint. Indeks diperlukan misal pada `leads(triage_status)`, `tickets(status)`, `invoices(due_date)`. Token warna (di UI) dan env config tidak ditulis ke DB.

Workflow Transaksional (Batasan Atomic)

Setiap perubahan penting diproses dalam satu **transaksi atomik** (misal via RPC/Function di Supabase). Contoh workflow:

- **CRM – Lead Lifecycle**:
- **Create Lead** (status = New) masuk tabel `leads`. Diadakan trigger: buat entri audit. (Limitation: minimal data validasi jika kolom kritis kosong → reject).
- **Triase Lead**: Marketing dapat mengubah `triage_status: New → In Review`. Status ini dikunci oleh peran (hanya marketing staff/manager). Otomatis buat task aktivitas call/email berikutnya pada tabel `activities`.
- **Qualify Lead**: Marketing ubah `triage_status: In Review → Qualified`. Saat ini, backend **otomatis mengkonversi lead menjadi opportunity**: membuat baris di `opportunities` (dengan FK `lead_id`) dan, bila perlu, data di `contacts`/`accounts`. Pada saat yang sama, kirim notifikasi atau reminder ke sales (dapat melalui pembuatan `activities`). Transaksi memastikan *semua* entri baru dibuat (lead tetap tercatat, atau bila di-archive status non-aktif).
- **Handover ke Sales**: Begitu lead ter-Qualified, record yang sama muncul di **Sales Inbox** (lihat Mapping di bawah). Sales dapat "claim" lead: proses POST ke `/api/crm/leads/{id}/claim` mengubah kolom `sales_owner_user_id = current_user` dan membuat aktivitas follow-up. Lead yang di-claim langsung ditampilkan di **My Leads**.
- **Konversi ke Opportunity**: Saat sales membuat penawaran atau melakukan progress, record opportunity (baris di tabel `opportunities`) bergerak melalui *stage*: Prospecting → Negotiation → (Negotiation mungkin spawn tasks). Transisi *stage* dibuat atomik: misalnya move to "Won" (menutup deal) sekaligus menghitung revenue, membuat faktur AR, dan menutup kesempatan (menandai sebagai closed). Jika *stage* = Lost, simpan alasan `lost_reason`.
- **Akun Pelanggan**: Jika lead selesai (Won), data kontak/account dicek: jika belum ada, buat di `accounts`/`contacts`.

- **Consistency/Invariants**: Tidak ada "dead lead" tanpa pemilik; setiap lead/opportunity aktif **selalu punya owner** dan aktivitas next-action terjadwal. (Sesuai PDF SSOT, invariant lead/opportunity owner di setiap state).

- **Ticketing – Life Cycle**:

- **Buat Tiket**: Pengguna (masuk peran operasional atau sales support) meng-POST `/api/tickets` → status *Open*. Transaksi membuat entri audit.
- **Assign Tiket**: Admin/OPS dapat assign tiket (`/tickets/{id}/assign`) ke staff, status mungkin berubah ke *In Progress*.
- **Respon & Komentar**: Ketika staf membalas, status di-update (mis. *Need Response* atau *Waiting Customer*). Setiap komentar tersimpan di `ticket_comments` dan dicatat audit.
- **Perubahan Revisi**: Jika perlu klarifikasi lebih lanjut (kompetitor, negosiasi harga), status *Need Adjustment*. Detail `competitor_name` dan `competitor_cost` disimpan saat closing.
- **Resolve**: Staf menandai selesai (*Resolved*) dan memberikan outcome (beserta optional penjelasan). Disini RPC menangani generate entri penutupan (jika ada proses bisnis lanjutan).

- **Close**: Supervisior lalu menutup tiket (*Closed* dengan *close_outcome*: won/lost). Transaksi penutupan memastikan semua data (tiket, komentar, lampiran) konsisten, dan satu catatan audit final dibuat. Setiap status dijaga agar perpindahan sesuai alur (tidak ada status terlewat tanpa log).

- **DSO/AR Workflow**:

- **Generate Invoice**: Ketika penjualan selesai (dari CRM apabila dibutuhkan), dibuat entri `invoices` (status *Outstanding*). RPC memastikan `due_date` dihitung dari terms.
- **Update Pembayaran**: Saat pelanggan membayar, entri di `payments` dibuat dan `invoices.status` di-update ke *Paid*. Transaksi juga menghitung DSO (via query atau view). Jika lewat jatuh tempo, `invoices.status` menjadi *Overdue*.
- **Kontrol DSO**: KPI/AR rutin menghitung DSO harian (via fungsi RPC) untuk memantau performa. Seluruh perubahan kunci dicatat di `audit_logs`.

- **No Double Entry**: Data penjualan tidak di-input ulang di AR; hanya sebagai referensi `account_id`. Misalnya, invoice meminta *account_id* dari `accounts` CRM. Kebijakan database (foreign key) dan RLS menjamin data tunggal (no dupes).

- **KPI Engine**:
  Setiap perubahan data (misalnya lead baru, deal close, tiket response) ditangkap lewat event atau cron-job. Fungsi RPC rutin (daily) menghitung metrik seperti *jumlah lead baru per bulan, rata-rata response time tiket, konversi*, dll. Hasilnya dimasukkan ke `kpi_results`. Transaksi memastikan konsistensi data (misal perhitungan luas).

Pada setiap tahap di atas, **semua langkah penting dilakukan atomik** – bila salah satu operasi gagal, rollback. Misalnya, ketika *mengonversi lead ke opportunity*, harus berhasil menulis ke tabel `opportunities`, `accounts`, dan `activities` sekaligus, atau tidak sama sekali.

Integrasi Data Tanpa Duplikasi

Keempat modul berinteraksi via **kunci tak terduplikat**. Misalnya, data pelanggan (account) hanya disimpan di CRM; modul lain (Ticketing, AR) merujuk ke `account_id`. Jika staf tiket membuat permintaan terkait pelanggan, cukup referensikan `account_id` atau `lead_id` yang sudah ada – tidak membuat ulang data pelanggan. Ticket yang merujuk lead akan menggunakan FK ke tabel `leads`. Invoice AR merujuk `accounts` yang sama dengan data CRM. Dengan begitu tidak terjadi “double entry” data master.

Pembuatan data silang selalu menggunakan *lookup* by ID: misalnya saat lead di-Qualified jadi opportunity, tidak buat lead baru, tapi tetap gunakan `lead_id` yang sama. RLS dan constraint FK mencegah penulisan data yang duplikat atau tidak konsisten.

Rencana Tahapan Pengembangan (Tanpa Backtracking)

Pengembangan sistem dilakukan bertahap berurutan agar tidak ada perubahan besar mundur (iterasi linear):

1. **Setup Infrastruktur & Autentikasi** – *Scope*: Inisialisasi proyek Next.js 16 (App Router) dan Supabase Postgres (koneksi, RLS aktif). Buat migrasi tabel utama: `users`, `roles`, `departments`, + audit log. Definisikan 15 peran persis seperti di BLUEPRINT. *Alasan*: Landasan otentikasi dan otorisasi harus benar sejak awal [1].
2. *Output Teknis*: Migrasi SQL/Prisma, seed role dan user dummy. Konfigurasi env (SUPABASE_URL, KEY).
3. *To-Do*:
    - [ ] Konfigurasi Supabase proyek, aktifkan RLS untuk semua tabel utama [1].
    - [ ] Buat model `User` dengan field role sesuai blueprint.
    - [ ] Implementasi SSR Supabase client (menggunakan `@supabase/ssr`, cookie) [2] [3].
4. *Quality Gates*: Semua pengguna dengan role yang berbeda bisa login; RLS mencegah akses data antar-departemen (mis. ops tidak bisa lihat lead). API auth teruji (login/logout, refresh token).

5. **Modul CRM (Paling Utama)** – *Scope*: Fungsionalitas lead-to-account penuh.

6. Backend: Tabel `leads`, `opportunities`, `accounts`, `activities`, `lead_handover_pool`, relasi, serta RPC/trigger untuk state transition (convert lead, claim, dll).
7. Frontend: Halaman CRM (“Lead Inbox”, “Sales Inbox”, “My Leads”, “Pipeline”, “Activities”) sesuai SSOT.
8. *Alasan*: Data pelanggan dan penjualan adalah core bisnis. Harus stabil dulu sebelum modul lain.
9. *Output*: Skema DB CRM siap; REST/RPC endpoints (Next.js route) sesuai CRM spec PDF; UI halaman CRM (React).
10. *To-Do*:
    - [ ] Desain tabel CRM dan migrasi SQL/Prisma.
    - [ ] Implementasi RLS policy: e.g. marketing hanya lihat leads departemennya [1].
    - [ ] Buat endpoint CRM (leads, activities, convert) sesuai SSOT; implement RPC untuk workflow atomik.
    - [ ] Kembangkan halaman Lead Inbox, Sales Inbox, My Leads, Pipeline, Activities (komponen tabel, form triase, kanban).
    - [ ] Integrasi tombol navigasi “Claim”, “Qualify”, dsb dengan backend.
11. *Quality Gates*: Semua alur status (Lead: New→In Review→Qualified) berjalan otomatis; konsistensi invariant (setiap lead punya owner dan next activity); mapping status ke UI tepat (verifikasi dengan spesifikasi dokumen). Tidak ada data orphan. Semua route & schema sesuai SSOT CRM (tanpa deviasi).

12. **Modul Ticketing** – *Scope*: Fitur internal ticketing & quoting.

13. Sesuai schema dan flows di zip `ugc-ticketing-web-main`. Adaptasi status workflow (Open → … → Closed), API routes (tickets, assignments, quotes, SLA).
14. *Alasan*: Menentukan integrasi dengan data CRM (misal lead vs ticket) dan AR (quotes/invoice) pada tahap awal agar bisa diuji end-to-end.
15. *Output*: Tabel `tickets`, `ticket_comments`, `ticket_attachments`, `ticket_assignments`, `quotations` + migrasi RLS, API endpoints (Next.js Route). UI halaman Ticket List dan Detail.
16. *To-Do*:
    - [ ] Import/migrasi schema Ticketing (gunakan `supabase/migrations/*.sql` sebagai referensi).
    - [ ] Definisikan RLS policy (contoh: sales hanya lihat tiket departemen mereka).
    - [ ] Implementasi API `/api/tickets`, `/api/tickets/[id]/assign`, dsb.
    - [ ] Buat komponen UI: Ticket List dengan filter status/prioritas, Ticket Detail (tab komentar, lampiran, kuotasi).

17. *Quality Gates*: Alur status Ticket (Open → … → Closed) berfungsi sesuai blueprint kode (tanpa status terlewat). Status page mapping terverifikasi (lihat poin F). Error handling dan idempotency (mis. re-submit form tidak duplikat) teruji. Data tiket disimpan hanya sekali (tidak ada duplicate entry).

18. **Modul DSO/AR** – *Scope*: Pengelolaan invoice dan piutang.

19. Tabel `invoices`, `payments`, relasi ke `accounts`. API untuk buat invoice & catat pembayaran. Integrasi dengan modul CRM (gunakan account/customer ID) dan Ticketing (bila ada invoice terkait tiket).
20. *Alasan*: Penting untuk kalkulasi keuangan. DSO/AR bergantung pada data penjualan (CRM) dan keputusan tiket (misal lost sale).
21. *Output*: Schema AR & RLS, endpoint `/api/invoices`, `/api/payments`, UI untuk daftar invoice.
22. *To-Do*:
    - [ ] Desain/migrasi tabel AR. Pastikan kolom tanggal dan amount tepat; indeks di `due_date` untuk query aging.
    - [ ] Implementasi policy RLS: finance departemen saja akses tabel AR.
    - [ ] Endpoint API: buat invoice (otomatis hitung `due_date`), tandai bayar.
    - [ ] UI: halaman daftar invoice (terpisah by status), detail invoice (pay).

23. *Quality Gates*: Perhitungan DSO benar; tidak ada invoice tanpa referensi CRM. Pembayaran menyelesaikan invoice secara atomik. Laporan aging sesuai formula DSO.

24. **Modul KPI Engine** – *Scope*: Perhitungan KPI Marketing, Sales, SLA, AR.

25. Konsolidasi data (lead, opportunity, ticket) untuk menghasilkan metrik. Biasanya di-backend (RPC) dan hasil disimpan di `kpi_results`.
26. *Alasan*: Perlu data final modul sebelumnya agar KPI akurat.
27. *Output*: RPC kalkulasi KPI, tabel `kpi_results`, backend data pull untuk Dashboard.
28. *To-Do*:
    - [ ] Tentukan KPI (mis. conversion rate, avg. response time tiket, collection rate).
    - [ ] Tulis fungsi RPC untuk hitung tiap metrik.
    - [ ] Jadwalkan cron job (supabase Edge Function atau Next.js Cron) untuk periodic recalculation.

    - [ ] Endpoint API /supabase table untuk ambil hasil KPI.

29. *Quality Gates*: KPI sesuai definisi bisnis, validasi hasil dengan perhitungan manual. Output terupdate secara berkala.

30. **Modul Dashboard & UI Lainnya** – *Scope*: Visualisasi dan ringkasan (charts, KPI cards, report).

31. Berdasarkan data di modul lain: summary (total leads, pipeline value, invoices, etc), grafik tren.
32. *Alasan*: Input semua modul siap, bisa langsung tampilkan.
33. *Output*: Halaman Dashboard, Statistik Global, Charts. Komponen UI seperti Table, Card, Chart.
34. *To-Do*:
    - [ ] Desain UI Dashboard (sesuai referensi Dribbble & tema warna).
    - [ ] Implementasi komponen ringkasan (card angka, chart bar/line, filter periode).
    - [ ] Integrasi data via API (mis. fetch `kpi_results`, `tickets`, `leads`).
35. *Quality Gates*: Semua komponen responsif bekerja di semua breakpoint. Tema Light/Dark sesuai spesifikasi (perubahan warna bg/text mengikuti token) [4] [5]. Data dashboard konsisten dengan sumber (sum, count benar).

Setiap tahap tidak mundur; contoh, modul Ticketing tidak dimulai sebelum CRM siap, agar link data berjalan.

Status → Halaman Tujuan (CRM & Ticketing)

CRM: Berdasar spesifikasi SSOT, setiap status record diarahkan ke halaman yang sesuai:
- Lead status **New/In Review** → *Lead Inbox* (route `/crm/lead-inbox`, marketing queue).
- Lead Handed Over (handover_eligible tanpa owner) → *Sales Inbox* (`/crm/sales-inbox`). Sales Inbox menampilkan Handover Pool + overdue tasks [6].
- Lead Claimed (owner ditetapkan) → *My Leads* (`/crm/my-leads`). Menampilkan leads assigned ke user (diteruskan ke Opportunity Pipeline view).
- Opportunity aktif (stage Prospecting..Negotiation) → *Pipeline Board* (`/crm/pipeline`) – kanban/daftar peluang.
- Aktivitas yang dijadwalkan → *Activities Planner* (`/crm/activities`). Menampilkan semua aktivitas (calls, tasks) yang belum selesai.
(Halaman status CRM diverifikasi dari dokumen SSOT; tidak dibuat asumsi tambahan.)

Ticketing: Semua tiket (semua status) ditampilkan di *Ticket List* (`/tickets` halaman utama) dengan fitur filter status. Tidak ada halaman terpisah per status; misal klik filter “Open” menampilkan tiket dengan `status = Open`. Halaman *Ticket Detail* (`/tickets/[id]`) untuk melihat detail satu tiket berapapun status-nya. (Mapping ini bersumber dari implementasi repo Ticketing yang ada.)

Spesifikasi UI Detail (Komponen dan Interaksi)

Desain UI menggunakan **design system berbasis token**, mengikuti tema Light+Dark (warna utama aksen **#FF4600**). Sistem warna dibangun bertingkat: *elementary tokens* (warna dasar: putih, hitam, primer, sekunder, dll) → *semantic tokens* (error, success, accent, dsb) → *component tokens* (background card, btn

primary) [4] [5]. Light mode dominan latar putih, Dark mode latar gelap, text dan komponen beradaptasi secara otomatis via CSS variables.

- **Sidebar**: Navigasi vertikal kiri, kolom tetap. Terdiri dari ikon + label menu (CRM, Ticketing, DSO/AR, KPI, Dashboard). Di atas ada logo perusahaan. Sidebar bisa collapse (ikon saja) di mobile.
- **Topbar**: Bar horizontal atas, mencakup tombol toggle tema (sun/moon), notifikasi (opsional), dan menu profil user. Di mobile, topbar lengkap.
- **Tabel Data** (tabel datagrid): Header kolom dapat di-sort, filter, dan resize. Baris data memiliki zebra-striping. Setiap tabel (leads, tickets, invoices) memiliki pagination numerik. Komponen tabel memakai frame dengan warna background netral token (mis. bg-white / bg-gray-800). Baris klik mengarahkan ke detail.
- **Form & Input**: Label teks di atas input. Field wajib ditandai (*). Validasi inline (error merah di bawah field). Gunakan komponen UI konsisten (input text, select, date picker) dengan ukuran dan spasi seragam. Tombol submit primary berwarna aksen (#FF4600) dengan teks putih. Modal form (popup) ditampilkan tengah layar (scrim gelap di belakang).
- **Modal/Dialogs**: Untuk konfirmasi/hapus. Judul besar, pesan di tengah, tombol aksi (Cancel, Confirm). Modal responsif; di mobile melebar layar. Warna tombol menggunakan semantic: hapus=warning/destructive.
- **KPI Card**: Kotak ringkas (card) menampilkan angka utama (font besar), label, dan icon kecil. Background card light/dark sesuai tema. Contoh: *"Total Leads"* dengan angka 150. Warna icon/teks angka menggunakan warna primer/aksen.
- **Chart**: Diagram garis atau batang (bar chart) untuk tren waktu, pie chart untuk komposisi. Warna chart mengikuti palet tema (satu varian warna utama dan netral). Ukuran chart responsif.
- **Grid Responsif**: Menggunakan grid 12-kolom. Breakpoint tipikal: ekstra-kecil (<500px): 1 kolom; kecil (500–768px): 2-3 kolom; sedang (768–1200px): 4-6 kolom; besar (>1200px): 12 kolom [7]. Misal Dashboard pada desktop: 4 card/row, pada tablet 2/row, di mobile stack 1/row. Grid beradaptasi sesuai prinsip responsive design [7].
- **Tema Gelap/Terang**: Toggle tema langsung mengganti variabel warna (background, teks, border). Semua komponen (sidebar, tabel, form, card) otomatis berubah melalui CSS variable token. Misal teks normal berubah dari hitam (#000) ke putih (#FFF) di Dark. Interaksi (hover, focus) punya warna highlight (#FF4600 lebih terang/donar) agar kontras.

Panduan Pengembangan (Best Practices)

- **Supabase SSR Pattern**: Gunakan dua client Supabase: *browser client* di komponen React (CSR) dan *server component client* di SSR/Route Handler menggunakan `@supabase/ssr` [2] [8]. Buat utilitas `lib/supabase/client.ts` dengan `createBrowserClient(...)`, dan `lib/supabase/server.ts` dengan `createServerComponentClient(...)`. Tambahkan *Proxy* (nextjs middleware) untuk auto-refresh auth token [9]. Jangan taruh logika auth di komponen server langsung tanpa `getClaims()` [10].
- **Access Pattern & RLS**: Aktifkan RLS di semua tabel penting [1]. Tulis policy sederhana: misal *"user dapat baca akun mereka sendiri"* menggunakan `auth.uid()`. Pisahkan schema per modul bila perlu, tapi RLS utama cukup gunakan kolom `department` / `role` di user. Contoh: tabel `leads` punya policy: marketing cuma baca leads di departemennya; sales cuma baca leads handover (owner).

- **Error Shape Standar**: Kembalikan error dalam format konsisten, misal `{error: { code: string, message: string }}`. Endpoint Next.js API hendaknya pakai error boundary dan selalu response JSON. Gunakan status HTTP tepat (400, 401, 403, 500).
- **Idempotency**: Untuk API yang berisiko double-submit (mis. create lead, create ticket), terapkan idempotency key di header. Simpan key tersebut di DB atau Redis, cek di awal request agar operasi kedua dengan key sama diabaikan atau di-return data lama.
- **Logging / Audit**: Setiap aksi CRUD penting menulis ke `audit_logs`. Bisa di-handle via trigger SQL (INSERT di audit_logs setiap insert/update/delete) atau di backend sebelum commit. Log wajib berisi `user_id`, aksi, keterangan perubahan.
- **Seed Data untuk QA**: Sediakan seed minimal di DB (via Prisma/migrations). Contoh: 10 sample leads, 5 opportunities, 3 tickets, 5 invoice, 15 user dengan variasi role. Agar QA dapat menguji fungsional tanpa setup manual.
- **Teknikal Lain**: Ikuti arsitektur Next.js App Router (folders > route.tsx). Struktur project modular (mis. `src/lib/`, `src/components/`, `src/app/(dashboard)/tickets/`, dsb). Pastikan semua API route dalam `src/app/api/**/route.ts`. Gunakan TypeScript tipe kuat untuk props/UI.

Modul Spesifik

CRM

Implementation Guide:
1. **Desain Data**: Buat migrasi tabel CRM (leads, opportunities, accounts, activities, lead_handover_pool). Definisikan enum `triage_status` (New, In Review, Qualified) dan `stage` (Prospecting, Negotiation, Won, Lost, Dismissed).
2. **Policies & RLS**: Tuliskan RLS di Supabase (mis. hanya marketing dept dapat insert ke `leads`; hanya sales dept dapat claim).
3. **Endpoints API**: Develop API routes Next.js:
- `GET /api/crm/leads?view=inbox` → lead inbox query.
- `POST /api/crm/leads/{id}/claim` → claim lead.
- `POST /api/crm/leads/{id}/convert` → convert ke opportunity.
- `GET /api/crm/opportunities`, dsb.
4. **UI Pages**: Kembangkan halaman dengan App Router: `/crm/lead-inbox`, `/crm/sales-inbox`, `/crm/my-leads`, `/crm/pipeline`, `/crm/activities`. Integrasikan komponen tabel, kanban, form triage. Gunakan state management minimal (bisa SWR/RPC fetch).
5. **Automation & Cadence**: Setup event triggers: saat lead ke Qualified, auto panggil API convert dan notifikasi.

To-Do Checklist:
- [ ] Migrasi tabel CRM siap.
- [ ] RLS policy tertulis & diuji (marketing, sales, support).
- [ ] API CRM implement dan teruji (Lead create, update status, convert).
- [ ] UI Lead Inbox (filter New/In Review) & Sales Inbox (filter handover_eligible).
- [ ] Fungsionalitas “Claim Lead”, “Qualify Lead” bekerja.

Quality Gates:
- Semua status sesuai SSOT (New→In Review→Qualified→Opportunity) berjalan otomatis.

- Role marketing tidak bisa akses data sales (dan sebaliknya).
- Tidak ada lead tanpa owner/aktivitas berikutnya.
- Halaman status mapping validasi (misal, lead *Qualified* harus hilang dari Lead Inbox).

Ticketing

Implementation Guide:
1. **Desain Data**: Migrate tabel Ticketing berdasarkan schema di zip (tickets, ticket_comments, ticket_attachments, ticket_assignments, quotations). Pilih enum untuk status, priority, type.
2. **Policies & RLS**: Pastikan RLS: e.g. hanya user di dept terkait atau super_admin bisa read/write tiket.
3. **Endpoints API**: Implement endpoint Next.js:
- `GET /api/tickets` (list dengan filter query),
- `POST /api/tickets` (create),
- `PUT /api/tickets/{id}` (update status, details),
- `POST /api/tickets/{id}/assign`, `/attachments`, `/comments`.
4. **UI Pages**: Buat halaman Ticket List (`/tickets`), dengan filter status/prioritas (komponen select dan search). Buat halaman Ticket Detail (`/tickets/[id]`) menampilkan info tiket, komentar, lampiran.
5. **Workflow**: Pastikan status flow sesuai: tombol "Reply" atau "Close Ticket" update status sesuai logika (Open→Need Response→Resolved→Closed).

To-Do Checklist:
- [ ] Migrasi schema Ticketing tepat sesuai blueprint (cek `schema.sql`).
- [ ] RLS policy dibuat untuk table tickets.
- [ ] Endpoint create/update tiket teruji (status berpindah sesuai alur).
- [ ] UI Ticket List lengkap (filter status), tombol navigasi ke Detail.
- [ ] Fitur komentar & lampiran dihalaman detail.

Quality Gates:
- Status-transisi ticket persis seperti pada spesifikasi (tidak ada state orphan).
- Filtering status di UI match dengan backend (e.g. filter "Waiting Customer" benar menampilkan status `waiting_customer`).
- Tidak ada duplikasi tiket atau komentar ganda.
- Email/SLA reminders (jika ada) berfungsi sesuai harapan.

DSO/AR (Finance)

Implementation Guide:
1. **Desain Data**: Buat tabel `invoices`, `payments`, `ar_aging` (optional view). Enum status invoice.
2. **Policies & RLS**: Finance (role finance) dapat lihat/ubah, other dept read-only atau none.
3. **Endpoints API**:
- `POST /api/invoices` (buat invoice),
- `GET /api/invoices`, `PUT /api/invoices/{id}/pay`.
4. **UI Pages**: Halaman Invoice List (`/ar/invoices`): tabel invoice dengan filter status (Outstanding, Paid, Overdue). Halaman detail invoice (Lihat, tandai bayar).
5. **Integrasi**: Buat trigger atau RPC yang menghitung DSO ketika invoice overdue.

To-Do Checklist:
- [ ] Migrasi tabel AR & indexing `due_date`.
- [ ] RLS policy finance-only.
- [ ] API untuk create/pay invoice terpasang (termasuk validasi total vs pembayaran).
- [ ] UI daftar invoice responsif.

Quality Gates:
- Invoice hanya terkait account yang valid.
- Pembayaran meng-update status tanpa error.
- Perhitungan DSO di QA (mis. rumus DSO benar).

KPI Engine

Implementation Guide:
1. **Desain Data**: Tabel `kpi_config` (definisi KPI), `kpi_results`.
2. **Calculations**: Tulis fungsi RPC/Edge Function untuk hitung:
- *Marketing*: Total leads, Qualified rate, dll.
- *Sales*: conversion rate, win rate.
- *Ticketing*: avg response time, tickets closed.
- *AR*: Collection rate, DSO.
3. **Cron Job**: Jadwalkan perhitungan harian (Gunakan Vercel Cron atau Supabase Edge).
4. **Endpoints**: API baca hasil KPI (bisa ke tabel kpi_results via RPC/Edge).

To-Do Checklist:
- [ ] Tentukan KPI list & definisi (sesuai permintaan stakeholder).
- [ ] Implement fungsi perhitungan (SQL/RPC).
- [ ] Setup scheduling (cron daily).
- [ ] Testing dengan data contoh.

Quality Gates:
- Hasil KPI sesuai ekspektasi (uji manual).
- Tidak ada duplikasi perhitungan (gunakan key periode).
- Perhitungan cepat dan akurat.

Dashboard (UI)

Implementation Guide:
1. **Desain Layout**: Sketsa page summary: beberapa KPI card di atas, grafik tren, dan tabel ringkasan.
2. **Komponen**: Buat komponen `Chart` (mis. Chart.js), `CardKPI`, dan tabel summary.
3. **Integrasi Data**: Fetch data via API: KPI (cards), leads/opps (tabel ringkasan), invoices (grafik aging).
4. **Responsiveness & Theme**: Pastikan semua elemen mengikuti grid responsif dan token warna.

To-Do Checklist:
- [ ] Implement halaman Dashboard di `/dashboard` (menggunakan Suspense/SSR).
- [ ] Desain komponen `KpiCard`, `LineChart`, `BarChart`.

- [ ] Pastikan layout grid sesuai breakpoint (2 kolom di tablet, 4 di desktop) [7].
- [ ] Toggle theme di UI Dashboard (cek kontras).

Quality Gates:
- Data di dashboard akurat sesuai data modul (cross-check nilai).
- UI konsisten, tidak ada overflow di mobile, semua chart dibaca mudah.
- Tema gelap terang berfungsi sempurna; warna aksen utama #FF4600 tampil kontras [4] [5].

Daftar Artefak Akhir

- **SQL Migrations**: daftar file migrasi (mis. `00001_init.sql`, `00002_rls.sql`, dst untuk semua modul).
- **RPC/Function List**: nama fungsi RPC penting (mis. `convert_lead_to_opportunity()`, `calculate_dso()`, `compute_kpi_sales()`, dll).
- **API Routes**: Ringkasan endpoint (mis. `GET/POST /api/crm/leads`, `/api/tickets`, `/api/invoices`, `/api/kpi`, dll).
- **UI Komponen Reusable**: (mis. `Sidebar`, `Topbar`, `DataTable`, `Modal`, `KpiCard`, `Chart`, `ThemeToggle`).

Dokumen ini mengikuti struktur Markdown agar mudah dipindah ke Notion. Semua spesifikasi detail, logika, dan alur telah diselaraskan dengan sumber primer (BLUEPRINT dan CRM SSOT) untuk menjadi satu sumber kebenaran sistem. Referensi teknis tambahan (SSR Supabase [2], warna token [4] [5], responsif grid [7], dan RLS [1]) dijadikan landasan agar implementasi konsisten, aman, dan tepat sasaran.

---

[1] Row Level Security in the Next.js Supabase Turbo kit
https://makerkit.dev/docs/next-supabase-turbo/security/row-level-security

[2] [3] [8] [9] [10] Creating a Supabase client for SSR | Supabase Docs
https://supabase.com/docs/guides/auth/server-side/creating-a-client

[4] [5] Color tokens: guide to light and dark modes in design systems | by Victoria Serebrennikova | Bootcamp | Medium
https://medium.com/design-bootcamp/color-tokens-guide-to-light-and-dark-modes-in-design-systems-146ab33023ac

[6] California Enterprise Architecture Framework Views
https://cdt.ca.gov/services/wp-content/uploads/sites/2/2020/10/CA-Enterprise-Architecture-Framework-Views.pdf

[7] Breakpoints in Responsive Design - NN/G
https://www.nngroup.com/articles/breakpoints-in-responsive-design/