INFIBIZZ / ENGINEERING HANDOFF

InfiBizz

Project documentation & backend handoff

Tab-wise features • Data model • API contracts • Business rules • Acceptance tests

Prepared for Mohit Bansal

Version 1.0 · 30 October 2026

Source-based specification of the current project. Recorded, derived, local and illustrative functionality are distinguished throughout.

16 application tables · 28 ordered migrations · Owner dashboard + private partner portal

Contents

1. Scope, product and reading guide 3

2. Navigation, actors and access 4

3. Overview tab 6

4. AI Advisor tab 7

5. Partners tab and setup 8

6. Lead-to-payout and pipeline 10

7. Customers tab 11

8. Billing & Payments tab 12

9. Campaigns tab 13

10. Offers & API and public intake 15

11. Partner portal tab-wise specification 16

12. Backend architecture and service boundaries 18

13. Calculation catalogue and backend invariants 19

14. Security, audit and concurrency requirements 20

15. Backend implementation sequence and acceptance tests 21

16. Open work, limitations and decisions needed 23

17. Exact public API compatibility contract 24

18. Internal operation contracts 26

19. Supplemental feature details 28

20. Application data dictionary 29

21. Schema amendments and database routines 38

22. Source inventory and handoff notes 41

1. Scope, product and reading guide

1.1 Product purpose

InfiBizz is a lending-business operations dashboard for a business owner working with LSPs, DSAs, merchants and other partners. It manages partner agreements, customer lead/application stages, partner-specific credit policy checks, campaign spend and lead reporting, uploaded billing documents, earned payouts and actual payment records. A separate partner portal allows invited partners to submit leads, record campaign figures, set their own platform limits and report payout receipts.

The main owner tabs are Overview, AI Advisor, Partners, Billing & Payments, Customers and Campaigns. Campaigns is served at /ads; the former Campaigns route is a compatibility redirect. There is no separate main Payments or Credit Criteria tab.

1.2 Status vocabulary

• RECORDED: Owner- or partner-entered records persisted in Lovable Cloud. Recorded does not mean independently verified by a bank, bureau or ad provider.

• DERIVED: Computed from recorded rows, such as recorded spend / leads, contractual payout estimates and outstanding balances.

• LOCAL: Stored only in the current browser or resettable sample workspace; not a shared backend record.

• ILLUSTRATIVE: Seeded historical charts, default funnel assumptions, modeled budgets or synthetic tracking days.

• NOT CONNECTED / BLOCKED: External delivery, live ad imports and custom credit scoring that require missing configuration or business inputs.

• PROPOSED: Recommended backend implementation or hardening; not a claim that it exists today.

1.3 Important boundaries

• A lead is not an application; an application is not an approval; an approval is not a disbursement.

• A contractual payout estimate is not an earned payout snapshot. An earned payout is not payment proof.

• Aggregate campaign lead totals are not customer records and must not create earnings or payments.

• Downloads do not imply leads or revenue. Ad budget limits govern recorded entries, not actual Google or Meta delivery.

• Never manufacture revenue, partner policies, credit probabilities, recorded dates or external delivery events.

• Database migrations and pure rule modules are authoritative for implementation details; reconcile later migrations before recreating the schema.

2. Navigation, actors and access

2.1 Owner navigation map

Tab / view	Address	Responsibility
Overview	/	Today, forecast, recorded operations, reminders and embedded insights
AI Advisor	/advisor	Editable business data and streamed analysis
Partners business	/partners	Partner business, funnel, pipeline and lead-to-payout path
Partner arrangements	/partners#details	Saved agreements, intake links and portal invitations
Partner credit criteria	/partners#criteria	Partner-specific bureau and DPD configuration
Partner onboarding	/partner-onboarding	Agreement, criteria, LSP payout plan, budgets, invite
Individual partner performance	/partner-performance/$code	One saved partner's cohorts and economics
Customer listing & stats	/customers	Filtered planning list and customer analytics
Customer onboarding	/customers#onboarding	Lead → application → approval → disbursement
Billing & payments	/billing and /billing#payments	Invoice/receipt ledger, earnings and actual transfers
Campaigns: Google	/ads#google	Owner + partner Google spend/leads, CPL, budgets
Campaigns: Meta	/ads#meta	Owner + partner Meta spend/leads, CPL, budgets
Plan & run	/ads#campaign-tools	Browser-local campaign workspace
Analysis	/ads#analysis	Spend trends, attributed outcomes and comparisons
Offers & API	/ads#partner-offers	LSP offers, redirect links, keys and callbacks
Message rates	/ads#rates	Browser-local cost assumptions per message

Compatibility: /campaigns redirects to /ads, preserving the incoming hash or using #campaign-tools. /payments redirects to /billing#payments; /criteria to /partners#criteria; /today to /#today; /insights to /#overview-insights. /partner-deals, /partner-policies, /funnel and /economics also exist as supporting or legacy content routes; preserve their actual behavior when porting navigation.

2.2 Actors and permissions

Actor	Allowed behavior	Forbidden behavior
Business owner	CRUD own deals, customers, documents, reminders, spend and budgets; invite/revoke partners; record earnings/payments	Read or modify another owner's private records
Invited partner member	Read enabled memberships; own partner leads, spend, budgets, earnings and receipts; own contact profile	Read owner-wide customer/ad records; self-assign a deal or owner; change owner caps
Public intake visitor	Read minimal partner name and submit a lead with valid token and consent	List leads, select an owner, approve leads or log payouts
Partner integration	Key-authorized offer metadata, tracked redirect and click-bound callbacks	Submit arbitrary owner/customer associations or fabricate earned payouts
Sample workspace user	Manipulate fictional in-memory records and reset them	Cause database, file-storage, messaging or real payment writes

Authentication is required on protected operations, not merely at page navigation. Source implements owner session-based data access and partner email/password sign-in. Partner invite acceptance requires a confirmed email matching the invitation. Hosted signup/sign-in provider settings were not inspected for this documentation.

2.3 Shared controls and sample mode

Owner shell includes partner, tenant and channel filters, plus Sample data and Reset sample controls. Shared filter state is not a universal backend query: each view applies only the filters it supports. Sample mode defaults on unless browser storage says off. Selection uses infibizz-sample-mode; fictional records are kept in isolated memory. Reset clears/recreates customers, deals, documents, reminders, earnings, payments, campaign budgets and owner ad logs. Public intake and the partner portal bypass owner/sample providers.

3. Overview tab

3.1 Today and week-ahead forecast

TodayPanel shows local-day figures for leads, approvals, partner payouts and spend, including separate partner ad spend, owner logged Meta spend and local message-campaign spend. It includes 28-day trend charts and a week-ahead projection. Partner leads are identified from recorded portal/intake lead sources, not invented arrivals.

Inputs include customer stage dates, partner_campaign_costs, spend_entries, browser campaignDaily results and partner_payouts. Local-day labels and UTC date arithmetic must be handled deliberately: timestamps represent instants; business stage/spend dates are date-only facts.

Forecast is the trailing seven calendar days' average multiplied by seven; zero-event days remain part of the basis. Hide the forecast when there is no recent basis. It is an extrapolation, not a guarantee and not a provider forecast. Keep the source-specific spend lines separate so local estimated costs cannot be mistaken for platform bills.

3.2 Recorded business overview

Refresh reads customers, saved partner deals, billing documents and reminders. Display customer counts, last-week/last-month onboarding, applications, approvals, disbursements, disbursed amount, conversion, expected partner payouts, billed/paid/outstanding/overdue amounts and per-partner snapshots. Alerts link to relevant customer, partner or billing work; Remind me copies the alert title into the reminder form.

Data service requirement: return owner-scoped facts or selectors, including source freshness and empty states. Do not use seeded counts when real records are absent. The dedicated recorded Overview uses real selectors; the embedded DashboardView insights still contains illustrative historical/economics/ad/campaign metrics. A production reporting backend should label or replace those explicitly.

3.3 Reminders

Fields: title (trimmed 1–200 characters), local date/time converted to a UTC due_at instant, priority HIGH/NORMAL/LOW and nullable completed_at. Actions: create, list Open/Completed/All, snooze one day, complete/reopen and delete. Reminders are private to the owner. Due checks run while Overview is open; no external email, push or scheduled notification delivery is implemented.

Backend needs CRUD, ordered due queries and owner checks. Proposed hardening: audit snooze/completion, clarify daylight-saving semantics for “one day,” and add notification jobs only after delivery providers and user preferences are configured.

4. AI Advisor tab

4.1 Features and input

Editable datasets: Partners, Campaigns, Billing and Ads. Owner can enter/paste data, ask a question, start analysis and stop a streamed request. Default text uses fixture data; sample mode rebuilds parts of Partners and Billing. The advisor is not a guaranteed live export of all owner records. Output is Markdown organized into Key trends, Risks & red flags, Recommended next steps and Watch next month.

The advisor does not save recommended actions into deals, budgets, customer decisions or payments. Treat its conclusions as advice requiring source verification.

4.2 Existing HTTP behavior

POST /api/advisor accepts JSON with data: a map of text values and optional question. Invalid JSON or no non-empty text gives HTTP 400. Missing AI configuration gives HTTP 500. Each filled dataset value is truncated to 20,000 characters and the question to 1,000. Response is streamed UTF-8 plain text; cancellation uses request abort. Stream failures append [[ERROR]] followed by a readable message. Rate-limit, credits and access-denied failures have distinct messages.

Credentials and the AI call stay server-side. Source endpoint does not apply the same explicit owner middleware as protected record functions; do not assume UI access protects it. Proposed production hardening: endpoint authentication, request/body limits, per-owner rate limiting, abuse protection, no-store for sensitive responses, input validation, and provenance-aware datasets. Never let AI overwrite source facts or issue financial mutations autonomously.

5. Partners tab and setup

5.1 Business with partners

The business subtab presents saved partners alongside recorded leads, applications, approvals, disbursements and contractual payout estimates. Refresh re-reads customers/deals. Customer partner codes without matching saved agreements are surfaced for resolution; do not silently create a partner. Partner names open individually scoped performance pages.

Detailed panels include policy status/reasons, follow-up counts, recent customers, live lead feed, monthly lead cohorts, lead-to-payout path, full pipeline and partner economics. Expected payout rows include base, GST, total, billable events, payment days, missing percentage amounts and events outside agreement dates. Unknown net income and ROI are not fabricated.

5.2 Partner details & arrangements

Agreement fields: name, stable uppercase code, type LSP/DSA/Merchant/Other, payout model CPL/CPA/CPD, kind FLAT/PERCENTAGE, rate, optional minimum and maximum payout, GST-inclusive flag, agreement start/end dates, payment days, notes, campaign budget and spend limit. Code is unique per owner and read-only when editing; preserve attribution links by keeping it stable.

Validation: trimmed non-empty name/code; CPL only FLAT; finite non-negative rates, percentage ≤100; non-negative min/max with min ≤ max; payment days non-negative integer; end date not before start. Budget and cap are optional, non-negative, at most ₹1 billion; if both provided, budget ≤ cap. Null cap means uncapped, while zero is a real zero allowance.

5.3 Partner-specific credit criteria

Each saved deal stores nullable bureau_min, bureau_max and dpd_max. Bureau bounds are integers 0–900 and min ≤ max; DPD maximum is a non-negative integer. Check only customers assigned to that partner. Inclusive comparisons produce PASS, FAIL, REVIEW (required values missing) or UNCONFIGURED (no limits). A known failure takes precedence over missing information.

Derived checks do not overwrite recorded application, approval or disbursement dates. Only verified policy passes contribute to eligible CPL estimates. This is a configured threshold check, not a credit bureau integration and not an approval probability.

5.4 Partner onboarding wizard

Six steps: identity; agreement dates and notes; credit criteria; LSP payout arrangement; campaign budget/spend limit; review and invite. Collect partner name/code/type and invited email, then agreement and policy fields before creating portal access. The LSP plan refers to the payout arrangement, not an unrelated product offer.

Validate the deal, policy and budgets before save. Save one deal before inviting; retrying an invitation reuses the saved deal rather than inserting a duplicate. Real invitation generation returns a shareable link with a seven-day claim expiry; no invitation email is sent by this flow. Sample mode saves only a fictional deal and never sends/creates a real invite.

5.5 Intake links and portal access

Owner creates/rotates or disables an intake link for a saved deal. Store only a hash of the random token; reveal the raw token/link on creation only. Disable must block future submissions. Portal invitations bind saved deal, owner and normalized invited email. Reinvite rotates token and clears member binding; revoke disables access. Membership must remain enabled for every read/write.

5.6 Individual partner performance

/partner-performance/$code scopes all customers, policy outcomes, stage counts, cohort rows, contractual payouts and payment summaries to the selected owner's saved partner code. Shared owner filters must not broaden or replace the selected partner. Missing/unavailable partner is an explicit empty/error state. Cohorts are based on lead dates; stage completion is based on recorded stage dates.

6. Lead-to-payout and pipeline

6.1 Guided owner path

Select a saved lead, assign a saved partner, record application, then approval date and amount. View earnings and linked payment events with effective date, log timestamp, amount and recorder (Owner or Partner). Refresh re-reads payout events; Billing link opens #payments. New lead opens customer onboarding.

Partner reassignment locks after application, approval, disbursement, an offer association or any earned payout. The database enforces the category of rule, not just a disabled selector. Earnings must match the customer's assigned saved deal and owner. Application is required before approval; later dates cannot be invented by progressing the UI.

6.2 Pipeline reporting

Show lead origin (capture source), channel, campaign, assigned partner, current stage, lead/application/approval/disbursement dates, elapsed approval/disbursement time, expected or recorded earning, due date, paid and remaining. Recorded earned snapshots take precedence over estimates. Due dates derive from recorded earning/terms when appropriate; unknown dates remain unknown.

Only actual payments linked by earned_payout_id reduce the lead balance. Generic contract transfers are tracked separately and cannot be arbitrarily allocated to a lead. Partial payments, unpaid, past-due and PAID states must share the same cents-accurate calculation with Billing and the portal.

7. Customers tab

7.1 Listing & stats

Listing is separate from onboarding. Planning statistics include total records, readiness, pending work, aging and priorities; customer analytics also include segment preview. Source is owner customer_applications or isolated sample rows. Records without recorded values must display unknown rather than inferred certainty.

Filters include name/search, partner, date range and date field, readiness, priority, age, stage, sort, lead channel, campaign and amount minimum/maximum. Channels include SMS, WHATSAPP, EMAIL, SOCIAL, GOOGLE_ADS, META_ADS, CALL, WALKIN and OTHER; “Not recorded” selects missing channel. Campaign options derive from distinct recorded partner_campaign values; “No campaign” selects missing attribution. Amount uses disbursed_amount if present, otherwise approved_amount; records with no amount are excluded whenever a range is set.

Backend query design: whitelist filter/sort values, use inclusive date boundaries, pagination and stable ties, owner scope before all filters, indexed partner/date/channel/campaign access. Current planning selectors are browser-side; a server replacement must preserve the same outputs and missing-value semantics, not introduce new partner rules.

7.2 Customer onboarding

Four stages: Lead capture, Application, Approval, Disbursement. New record can be created only at lead capture; existing records resume at their next stage. Inputs: customer name (1–200), optional mobile (≤30), saved partner code (≤100), optional state (≤3), bureau score integer 0–900, non-negative salary, DPD and activity days, shared date, optional application/approval/disbursement dates and optional approved/disbursed amounts.

Persist only selected-stage fields: capture writes identity/partner/profile/lead date; application writes date and supporting profile; approval writes approval date/amount; disbursement writes disbursement date/amount. Merge with saved record before validation. Do not write downstream stages implicitly.

Rules: application ≥ lead date; approval requires application and ≥ application date; disbursement requires approval and ≥ approval date. Approved amount requires approval date; disbursed amount requires disbursement date. Validate real calendar dates, finite non-negative amounts and saved partner ownership. Recalculate policy against that partner, without turning policy PASS into approval.

7.3 Live records and source attribution

Owner hook subscribes to protected customer changes with unique channels and cleanup, and re-fetches records on changes/focus rather than generating leads. Live state can be connecting/live/offline. Source attribution stores lead_source, lead_channel, partner_campaign and optional local campaign_id or partner_offer_id. Source capture, partner campaign name and browser plan ID are separate concepts; do not merge them into one guessed identifier.

8. Billing & Payments tab

8.1 Document workflow

Select invoice/receipt and upload PDF, PNG, JPEG or WEBP, at most 15 MB in the current UI. Store a private object in billing-docs under the owner's UUID folder, then an owner-scoped billing_documents row. Statuses include extracting, ready and failed; preserve error information. AI extraction downloads the authorized file server-side, extracts fields and writes ready or failed. Manual edit remains available after failure.

Fields include partner label, document number, receipt invoice reference, document/due dates, amount, GST, total and UTR. Open generates a five-minute signed URL; edit/save, re-extract and delete are supported. Deletion also removes the private file. Sample mode never uploads, opens real files or calls extraction.

Backend hardening (proposed): enforce size/MIME and file signature server-side, clean up orphan objects if metadata insert fails, serialize extraction, check update errors, validate extracted numeric/date fields and retain extraction provenance. File and database writes are not presently one transaction.

8.2 Invoice / receipt ledger

Only ready documents contribute. Match receipt invoice_ref to invoice doc_number after trim/lowercase normalization. paid per invoice is the sum of matching receipt totals; outstanding = max(0, invoice total − paid). PAID when outstanding ≤₹0.01, PARTIALLY_PAID when paid >0, otherwise RAISED. Overdue when remaining >₹0.01 and due_date is before today. Total paid is capped at each invoice total. Show unmatched receipts separately.

Existing matching uses normalized number, not a foreign key or partner-scoped match. Proposed backend: unique scoped invoice identifiers, explicit receipt allocation and overpayment review; do not silently change current ledger behavior without migration/reconciliation. Invoice ledger is separate from lead-linked partner payouts to avoid counting the same financial fact twice.

8.3 Earned payout tracker

Record a saved customer's earned payout for the assigned saved deal, with amount, earned_on and due_on. Earned record is unique per owner/customer. It is a snapshot of a recorded earning, not a bank transfer. Owner and partner recording paths share uniqueness and deal-assignment enforcement. Portal duplicate earning returns a readable already-recorded message.

8.4 Actual partner payments

Owner can log a transfer against a saved partner, or a linked payment against an unpaid earned payout. Linked form includes partner, earning, amount, date and UTR. Portal can report an actual receipt against the same earning. Shared records refresh Billing/pipeline at 15-second intervals and on focus; portal also refreshes every 15 seconds.

Atomic linked payment functions lock the earning, validate owner/member/deal, reject future dates and amounts above the remaining balance, and deduplicate retry submission UUIDs. UTR ≤60 characters. Save partial and full payments; use linked sums for customer balances. Generic contract payments reduce contract settlement, not an arbitrary lead balance. There is no bank API or actual fund transfer initiated by these log forms.

9. Campaigns tab

9.1 Structure and common tools

Google Ads and Meta Ads keep separate recorded reporting. Plan & run, Analysis, Offers & API and Message rates are shared once. One mounted campaign manager preserves draft state when switching subtabs. Per-platform model panels are collapsed and illustrative; they are not imported provider reports.

9.2 Google / Meta owner daily logs

Fields: platform google/meta, trimmed campaign name (1–80), valid logged_on date no later than today, spend (₹0–999,999,999,999.99), leads integer (0–2,147,483,647), optional revenue in the same money range. Save replaces the same owner/platform/campaign/day row through its unique key. Edit/delete history, refresh/poll/focus and inclusive From/To filters are provided.

Summary groups by exact campaign name within selected platform/range. Show spend, leads, CPL, revenue and ROI. Sorting supports lowest CPL, most leads or highest spend; unknown CPL belongs after known figures for efficiency comparison. Rows persist across reload in real mode; sample logs remain in resettable memory.

CPL = total spend / total leads when leads >0, else unknown. ROI percent = (recorded revenue − spend) / spend ×100 only if every selected daily row has revenue and spend >0; one missing revenue makes campaign revenue/ROI unknown. Loan disbursement amounts, expected contributions and partner payouts are not revenue.

9.3 Campaign budgets

Owner sets a lifetime budget by platform and campaign name. Unique owner/platform/campaign key; amount >0 and ≤999,999,999,999.99. Save/edit/delete, recorded spent, remaining, progress and over-budget warning. Spent uses lifetime owner_ad_campaign_logs only, not partner spend and not current date-filter spend. Remaining = max(0,budget − spent); over = max(0,spent − budget). This is planning/visibility, not Google/Meta enforcement.

9.4 Partner reporting and combined comparisons

Partner campaigns use saved deal scope and recorded partner_campaign_costs plus attributed customer rows. Display per campaign spend, reported/customer leads, approvals, disbursals, CPL and cost per approval. Missing spend stays unknown. Partner revenue is not recorded, so partner ROI remains unknown.

Combined owner/platform totals add owner daily logs and partner-scoped stats. “My campaigns vs partners” shows the owner alongside each partner's spend, leads and CPL. Sync partner totals re-reads records; it does not copy rows or write spend/leads. Repeated sync must not duplicate totals. Owner-wide logs cannot be displayed as a partner's private campaign data.

Reported daily ad leads replace corresponding attributed lead counts for the same partner/campaign/channel/date in aggregation; customer approval/disbursement counts remain recorded outcomes. Otherwise preserve attributed customer leads. Unsourced customer leads are excluded from campaign comparison and shown as unattributed where appropriate. Keep actual source rows for audit; aggregate totals never create customer records.

9.5 Plan & run

Plan includes name, channel SMS/WHATSAPP/EMAIL/SOCIAL, target segments PRIME/GOOD/NEEDS_REVIEW/NOT_ELIGIBLE, control percentage, fixed cost, start/end, frequency cap days and message. Social uses reach rather than the segment audience/control calculation. Preview estimates audience, sent/delivered/clicks/disbursals, cost, contribution, net and ROI.

Plans and lifecycle state currently use browser storage infibizz-campaigns-v1. Default segment sizes, delivery/click/disbursement rates and per-disbursal contribution are assumptions; day generation/tracking is simulated unless backed by explicit recorded inputs. Starting a campaign changes local status; it does not send messages or start an ad provider job.

Allowed moves: PLANNED → SCHEDULED or CANCELLED; SCHEDULED → RUNNING, PLANNED or CANCELLED; RUNNING → PAUSED or COMPLETED; PAUSED → RUNNING or COMPLETED; COMPLETED/CANCELLED are terminal. Frequency cap is a planning field, not guaranteed provider suppression.

9.6 Analysis and real spend tracker

Date range, campaign comparison, spend/lead trends, cost per lead/approval/disbursement, approval rate, CPL trend and suggestions. Real approvals/disbursals use customer campaign_id and recorded outcome dates, never simulated approvals. Local simulated days may still supply lead/delivery counts when messaging is disconnected.

Owner spend entries use the actual table spend_entries (not own_spend_entries): campaign_id, channel SMS/WHATSAPP/META, spent_on, amount >0 and notes. For a campaign with any logged spend, recorded entries replace all simulated spend; unlogged days count zero. Real spend does not make simulated lead counts real. LeadSourceTracker assigns channel/campaign to saved customers; unsourced leads are not guessed.

9.7 Message rates

Rates are rupees per message before GST, validated finite ₹0–₹100. Current defaults: SMS ₹0.18, WhatsApp ₹0.80, Email ₹0.05, Social ₹0.12. They are editable assumptions, not provider contracts. Saved to infibizz-channel-rates in browser storage; applying rates changes local forecast/analysis calculations. Backend migration should version rates per owner/campaign period rather than retroactively reprice historical actual bills.

10. Offers & API and public intake

10.1 Partner offers

Create a product offering linked to a saved LSP deal and campaign identifier, with website/app destination. Manage enabled state and generate/rotate/revoke an integration key; only its hash is stored. Partner API keys must remain server-to-server, never embedded in a public page/app bundle.

Tracked redirects append infibizz_click_id and campaign/source attribution to a saved HTTPS destination. Partner site/app must preserve the click ID and report a download or lead callback. No callback means no verified conversion. Callback ownership comes from saved offer/key/click, not from arbitrary payload owner/deal values.

10.2 Public intake

/intake/$token resolves only the saved partner name and displays a lead capture form. Fields include customer name, mobile, state, bureau, salary, DPD and explicit consent. It does not read owner customer lists. Hash token, verify enabled link and deal ownership, lock link, derive owner and partner association and insert a lead-only row with submission UUID deduplication. Disabled/missing links reject submission. Public routes bypass dashboard/sample mode.

11. Partner portal tab-wise specification

11.1 Sign-in and membership

/portal supports partner account creation and email/password sign-in; owner-issued invitation is accepted with its matching verified email. Profile fields never confer permissions. Multiple enabled memberships allow account selection; selection changes all partner-scoped views. Sign-out clears portal queries and form state. Missing/disabled access shows a no-active-access state rather than owner records.

Only Budgets currently has a persistent view query (?view=budgets); other portal tab selections are local UI state. Invitation is passed via ?invite=token. Treat tokens as sensitive; no-index/no-referrer is set on portal/intake views. Consider removing invite tokens from address history after claim as a proposed hardening.

11.2 Dashboard

Counts: leads, applications, approvals, disbursements, contractual expected payout, paid and outstanding. Includes Google/Meta campaign dashboard, payout tracker, actual receipts table, lead progress by recorded dates and monthly cohorts. Missing amounts produce an explicit warning. Partner rows returned to the display omit mobile, salary and bureau details even though scoped server calculations may use them.

11.3 Submit lead

Collect name, mobile, optional state/bureau/salary/DPD, channel, campaign and consent. Access ID and a retained submission UUID authorize/deduplicate the attempt. Server rechecks membership, derives owner and partner from the saved deal and creates only a lead. Retry retains the UUID; successful completion clears it and refreshes Dashboard. Partner cannot record an approval through this form.

11.4 Campaigns

Separate Google and Meta panels with date filters, totals, campaign comparisons and sorting. Daily totals form collects campaign name, date, spend and leads; same deal/channel/campaign/day replaces rather than adds. Display reported ad leads, real attributed approvals/disbursals, CPL and source-aware history: spend date, campaign, amount, leads, recorded_at and Business owner/Partner recorder.

Also includes all-channel campaign spend entry/history and attribution coverage; allowed lead channels include message/ad/call/walk-in/other. Use saved campaign names consistently. Non-ad append-only costs and daily ad totals are distinct entry types; avoid unintentionally counting both entries for the same bill.

11.5 Budgets

Google and Meta cards let partners save or remove their own lifetime platform spend limit (₹0–1 billion; null removes cap). Display current limit, recorded spend, remaining, utilization and over-limit warning. Total spend includes all their campaigns on that platform across all dates, not date-filtered figures. Owner per-campaign caps remain protected. Lowering a cap below existing spend may show overspend; downward corrections must remain possible.

Database uses deal locks for owner per-campaign/channel and partner per-platform checks. Every write path, including direct owner writes and portal daily replacements, must pass the same caps. A same-day replacement excludes the old amount from the projected total. Caps do not pause or change ad provider spending.

11.6 Payout tracker

Partner records an earned payout for one of their saved leads, then actual receipt amount/date/UTR against that earning. Show recorded earnings and linked paid/remaining balances. Owner's previously logged linked payment is visible and reduces available balance before partner entry. Atomic caps reject double counting/overpayment; retries share a submission ID. Earnings are not receipts, and forms do not transfer money.

11.7 Contact details

Own contact_name (≤200), phone (≤30) and company (≤200); save/load own profile. No permissions, role flags or deal IDs can be changed through a profile. Authentication lifecycle profile creation is separate from membership authorization.

12. Backend architecture and service boundaries

12.1 Current architecture

React 19 + TanStack Start v1, TanStack Router/Query, Tailwind v4, Lovable Cloud for private records/auth/storage. Browser code includes pure selectors, planning state, sample mode and authenticated owner queries. Internal authenticated functions handle onboarding, extraction, invitations and scoped portal operations. Public HTTP routes handle offers, redirect and conversion callbacks; Advisor is a separate streamed HTTP route.

Backend state must be durable, not held in server module globals. Authenticate each protected endpoint, derive owner from verified session, and scope before queries/mutations. Privileged reads/writes are limited to verified membership/public-token flows and service-only database functions; ordinary owner CRUD retains owner row policies. Authenticated responses must not be shared-cacheable.

12.2 Suggested service domains

• Identity & access: account/session validation, profile, invitation claims and revocation.

• Partner contracts & policy: saved agreements, stable codes, payout calculation and limits.

• Customer lifecycle: capture, attribution, ordered stage mutations, policy derivation and live changes.

• Campaign ledger: owner daily ad logs, partner costs/daily totals, budgets and source-aware summaries.

• Earnings & settlement: immutable earned snapshots, atomic linked payments and contract transfers.

• Documents: private file lifecycle, extraction jobs, validated metadata and ledger reconciliation.

• Offers: key management, click capture, safe redirects and conversion ingestion.

• Reporting & reminders: owner/partner scoped projections and reminder CRUD.

• AI: bounded data ingestion, streamed advice, provider errors and provenance.

These are logical boundaries for a replacement backend, not newly implemented services. A REST adapter may be added if needed; current createServerFn calls use framework serialization and are not generic JSON endpoints.

12.3 Entity relationships

Owner → many partner_deals, customer_applications, billing_documents, reminders, owner ad logs and budgets. Deal → intake link, portal memberships, partner costs, offers and payouts. Customer.partner_code is an owner-scoped stable code association, not an unconstrained global partner ID. Earned payout links customer and deal; linked payment references its earning. Offer click references saved offer; event references click; a lead event may create an owner-scoped customer. Profile identifies a user but does not grant membership.

Check each migration's actual FK/cascade before deletion design. Prevent losing financial audit trails when deleting a deal/customer; current cascades are not a complete retention policy. Proposed production approach: explicit archive/retention and immutable audit events, agreed with the business owner.

13. Calculation catalogue and backend invariants

13.1 Commercial payout

CPL bills only policy-passed lead dates within agreement; CPA bills recorded approval dates; CPD bills recorded disbursement dates. FLAT uses rate; PERCENTAGE uses each customer's approved amount for CPA or disbursed amount for CPD. Apply min/max per event before GST. Missing percentage amount is skipped and counted; event outside agreement is skipped and counted. Do not apply percentages to an average ticket when individual facts exist.

GST default is 18%. If exclusive: total = bounded base ×1.18. If inclusive: ex-GST base = bounded total /1.18; GST = total − base. Financial calculations should use decimal/cents, not accumulated binary floats; presentation may round INR while stored facts retain precision.

13.2 Payout balances

Lead paid = sum payments with that earned_payout_id; remaining = max(0,earned − linked paid). Contract paid = all saved deal transfers; outstanding = max(0,contract estimate − contract paid). Do not interchange these two balances. Payment concurrency must lock the same earning; replay of the same submission ID returns/retains the same receipt instead of adding it.

13.3 Campaign costs and ROI

CPL = sum spend / sum leads, not average daily CPL. Cost/approval and cost/disbursal require recorded denominators and spend; zero denominators are unknown. Owner revenue-based ROI requires complete daily revenue and positive spend. Partner ROI is unknown without partner revenue. Combined CPL uses combined spend and leads, not an average of owner/partner ratios.

13.4 Estimates and statistical results

Plan estimate: audience is selected default segment sizes (or social reach); control = rounded audience × control%; sent = audience − control; delivered/clicks/disbursals use channel assumptions; cost = sent × rate + fixed cost; modeled income = disbursals × ₹8,954. Those constants come from the guide/demo model, not verified earnings.

Two-proportion uplift uses treated/control disbursal rates and pooled standard error; absent group yields unknown. Tracking verdict: no disbursals → NO_CONVERSIONS_YET; negative net → STOP_OR_RETARGET; |z| <1.96 → INCONCLUSIVE; ROI ratio ≥3 → SCALE; otherwise CONTINUE. Without a real control group, attribution is not proof of incrementality.

Ad model: leads = a × budget^beta; disbursals = K × budget^epsilon; modeled profit = contribution × disbursals − budget. Optimal modeled budget applies only when 0<epsilon<1; SCALE next step is at most +30%. These should remain a separate modeled layer until calibrated from actual provider/cohort data.

13.5 Credit/segment limitation

Saved partner policies are genuine configurable thresholds. Legacy customer segment previews still use a provisional bureau-to-approval mapping (600→0, 850→100; missing bureau falls back to 40), illustrative conversion values and C1–C5 rules. These are not the owner's confirmed scoring formula. Do not implement them as a production underwriting or approval-probability API. Required inputs before custom calculator: variables, weights/formula, score range, cutoffs, missing-value handling, a worked example and approved chance method.

14. Security, audit and concurrency requirements

• Enable owner-only read/insert/update/delete policies and explicit grants for every private table; policies must validate both old and new owner on updates.

• Never accept user_id, owner_id, member_id or recorder identity from an untrusted payload as authority. Derive them from session or saved token/membership.

• Store token/key hashes only; use cryptographically random generation, rotation, revocation, no-store and bounded payloads. Raw keys are disclosed once.

• Portal privileged reads first verify enabled current-user membership and matching owner/deal; avoid widening owner-table access policies to partner users.

• Public endpoints expose no private lists. Verify API key/click linkage inside raw HTTP handlers; URL prefix or UI hiding is not security.

• Lock deals for all spend-cap checks and lock earnings for all linked payment checks. Replacement must subtract old row before adding new; downward corrections survive a lowered cap.

• Keep dates (effective date) separate from timestamps (logged_at); show recorder and log time. Existing recorder metadata is not a full append-only audit trail of edits/deletes.

• Protect files by owner folder and signed URLs. Enforce size/type at backend; never return private file paths as unrestricted URLs.

• Do not log raw credentials, tokens or sensitive customer financial fields. Minimize portal DTOs and API metadata.

• Proposed: provider signature checks, rate limits, fraud/duplicate detection, retention/archive policy, comprehensive audit events, extraction retry queue and transactionally consistent report snapshots.

15. Backend implementation sequence and acceptance tests

15.1 Recommended build order

1. Account/session validation, private profiles and owner scoping.

2. Apply full ordered schema plus indexes, constraints, grants, owner policies and date-order rules.

3. Implement partner deal/policy CRUD, stable codes and customer capture/stage mutations.

4. Add invitation/token issuance and membership claims/revocation; then scoped portal/intake capture.

5. Add campaign logs, partner daily replacement, recorded aggregation and all spend cap triggers.

6. Add earned snapshots, linked payment locks/replay protection and shared settlement read models.

7. Implement private documents, validated extraction and invoice receipt reconciliation.

8. Implement offers/keys/redirect/callback contracts and API abuse protection.

9. Implement owner/partner reports, live refresh, reminders and source-aware AI input.

10. Migrate browser-local plans/rates only if shared durable planning is required; add actual provider integrations separately.

15.2 Minimum acceptance checklist

• Two owners cannot read/write each other's deals, customers, logs, documents or reminders; changing payload IDs cannot bypass isolation.

• Two partner memberships cannot read/write another partner's costs/leads/payouts. Revocation immediately blocks protected actions.

• Wrong/unconfirmed invite email fails; same member can retry claim; different member cannot steal claimed invite.

• Intake/portal submission retry creates one lead; public read endpoints never expose lead rows. Disabled token rejects future submissions.

• Application before lead, approval without application and disbursement without approval fail through direct database writes as well as forms.

• Reassignment after application/offer/earning fails; earned payout must match assigned partner; one owner/customer earning survives duplicate attempts.

• Owner and portal concurrent ₹600 payments against ₹1,000 earning cannot both overpay; retry UUID does not add another receipt.

• Partial ₹400 payment on ₹1,000 earning yields ₹600 remaining in Billing, portal and pipeline after refresh/reload.

• Google and Meta daily logs stay separate; same campaign/day save replaces; ROI ₹1,500 revenue /₹1,000 spend = +50%; missing revenue or zero spend stays unknown.

• ₹900 spend and 30 leads gives ₹30 CPL; zero leads unknown; ranking does not average daily CPL.

• Partner daily reported leads and corresponding customer leads are not double counted; approvals remain recorded outcomes; sync does zero writes.

• Owner campaign cap and partner lifetime platform cap reject over-limit inserts/updates; same-day replacements and downward corrections behave correctly under concurrency.

• Budget ₹1,000 and lifetime spend ₹400 gives ₹600 remaining; lowering budget below spend shows overage; null cap differs from zero.

• Ledger matches case-insensitive trimmed invoice references, caps paid at total and flags missing/unmatched references without guessing.

• Failed extraction can be manually corrected; owner cannot open another owner's file; sample upload/messaging/payment actions perform no real writes.

• Forecast with no trailing data is hidden; zero days remain in the seven-day denominator; no external alert delivery is implied.

• API wrong/revoked key, paused offer, wrong click/offer and duplicate download/lead events fail or deduplicate according to contract.

15.3 Test and code map

Business/financial rules: bizz.test.ts, partner-deals.test.ts, partner-payouts.test.ts, earned-payouts.test.ts and portal-payments.test.ts. Customer lifecycle/reporting: customer-onboarding, customer-planning, customer-stats, partner-policy, partner-performance, lead-pipeline and lead-payout-path tests. Campaign rules: campaign-plan, owner-ad-campaigns, campaign-budget, campaign-comparison, partner-campaigns, partner-budget, portal-budget, portal-ad-totals and portal-campaigns tests. Capture/security: partner-intake, partner-portal and partner-offers tests. Overview/billing/sample: overview, daily-view, billing-ledger and sample-data tests.

Existing tests are implementation evidence, not proof of live external provider behavior. Run targeted unit tests, migration-level permission/concurrency tests and signed-in owner/partner end-to-end flows before release.

16. Open work, limitations and decisions needed

Area	Current status	Backend implication
Real SMS / WhatsApp delivery	Not connected; cards previously skipped	Add provider integration, consent/suppression, send jobs, delivery callbacks; do not relabel simulated tracking
Google / Meta provider imports	No connected live account feed established	Manual recorded logs are usable; provider spend imports and reconciliation are separate work
Credit calculator	Owner formula missing	Do not invent scoring/probability; replace provisional segment assumptions only after inputs supplied
Invitations and notifications	Manual share links; reminder delivery in app only	Configure verified sending and notification preferences before promises of email/push
Campaign plans/rates	Browser-local	No multi-device/shared plan consistency; durable migration is a separate requirement
Historical insights/ad models	Some seeded/provisional data	Label as illustrative or replace from facts; keep real reporting independent
Revenue / ROI	Optional owner daily revenue; partner revenue absent	Unknown is intentional; never use disbursed loan value as revenue
Financial verification	Recorded logs, not bank confirmation	Add reconciliation/approval workflows only if needed; logging does not move funds
Deployment/settings	Source inspected; hosted configuration not audited	Verify migration state, auth settings, storage policies, scheduled jobs and publication readiness

This PDF does not change the running app, apply migrations, connect providers or implement a replacement backend. The next build should preserve the current facts and contracts, then explicitly approve any new policy, architecture or reconciliation semantics.

17. Exact public API compatibility contract

17.1 Offer metadata

GET /api/public/partner-offers?offer_id=UUID. Authorization header: Bearer followed by a 64-character lowercase hexadecimal key. Hash the presented key and match an enabled saved offer. Invalid ID/key, disabled offer or mismatch returns HTTP 401 JSON {error: "Unauthorized"}.

Success HTTP 200 returns offer {id, name, website_url, app_url, campaign_id, campaign_name, channel}, website_tracking_url and nullable app_tracking_url. Return only these metadata fields; no private owner/deal/customer rows. Tracking URLs target the public redirect endpoint. JSON responses use Cache-Control: no-store. No browser CORS integration is established by this source contract; consumers should use their server.

17.2 Tracked destination

GET /api/public/partner-redirect?offer_id=UUID&target=website|app. Target defaults website. Success records a click and returns HTTP 302 Location: saved HTTPS destination, augmented with infibizz_click_id, utm_source=infibizz and utm_campaign from saved campaign_name. Headers include Cache-Control: no-store and Referrer-Policy: no-referrer.

Invalid query → 400 plain text “Invalid offer link.”; unavailable offer/click RPC result →404 “Offer unavailable.”; invalid destination →400 “Destination unavailable.” No key is needed for this public click link. URL must be saved, HTTPS, without embedded credentials or localhost; no caller-supplied arbitrary redirect destination.

17.3 Conversion ingestion

POST /api/public/partner-events?offer_id=UUID with the same Bearer key. Read JSON no larger than 16,384 characters (source counts text characters, not necessarily UTF-8 bytes). Strict schema rejects unknown event properties.

DOWNLOAD body: type="DOWNLOAD", event_id UUID, click_id UUID. LEAD body: type="LEAD", event_id UUID, click_id UUID and lead object. Lead requires customer_name (trimmed 1–200), mobile matching optional plus followed by 7–15 digits, nullable state (trimmed uppercase ≤3), nullable bureau_score integer 0–900, nullable monthly_salary finite ≥0, nullable worst_dpd_12m integer ≥0 and consent=true. Nullable fields must be included in the lead API object.

Responses: 401 {error:"Unauthorized"}; 413 {error:"Request too large"}; 400 {error:"Invalid JSON"}; 400 {error:"Invalid event or lead details"}; RPC rejection 400 {error:"Event rejected. Check access, click and event identifiers."}; success 200 {received:true,duplicate:true|false}. JSON uses no-store.

Atomic RPC binds click to offer. Same offer/event_id + same click/type returns duplicate; reuse with different click/type rejects. A second same-type event for one click is duplicate. One click can have one download and one lead. Lead event inserts a customer with lead_source=partner_api and policy_passed=false, attributed to the saved offer/campaign; it does not approve, earn or pay.

17.4 Example integration sequence

1. Partner server fetches offer metadata with a private key.

2. User follows returned website/app tracking URL; destination receives click ID.

3. Partner confirms an actual install/download or consensual lead and sends event with retained UUIDs.

4. On network retry, reuse the same event_id and click_id; read duplicate result instead of generating a new event.

5. Owner sees recorded offer clicks/downloads/leads; created lead enters Customers and the assigned LSP pipeline.

18. Internal operation contracts

These are framework server functions, not invented REST URL paths. GET/POST denotes the declared server-function method. Owner direct CRUD remains protected by private row policies; a replacement backend may expose equivalent operations with its own transport.

Operation	Input / return	Authority and persistence
extractDocument POST	{id}; {ok:true} or {ok:false,error}	Owner document and private file; writes extracted fields/status
saveCustomerOnboarding POST	{id:null or UUID,step:0..3,values}; saved row	Verified owner, saved partner, merged stage validation
manageIntakeLink POST	{deal_id,action:create/disable}; {token or null}	Owner deal; hashed link upsert/disable
getIntakePartner GET	token; {name} or null	Public narrow lookup of enabled hashed link
submitPartnerIntake POST	{token,submission_id,lead}; {received:true}	Service-only atomic token-bound lead insertion
managePortalAccess POST	{deal_id,email,action:invite/revoke}; {token or null}	Owner deal; seven-day normalized-email invite
listPortalAccess GET	no input; selected access rows	Current owner only; does not return raw tokens
claimPortalInvite POST	token; {accepted:true}	Verified matching email; conditional single-member claim
getPortalDashboard GET	no input; array of scoped memberships/reports	Enabled member → matching owner/deal; no owner-wide records
submitPortalLead POST	{access_id,submission_id,lead}; {received:true}	Member/access lock, lead-only creation
getPortalProfile GET	no input; contact fields or null	Own profile only
savePortalProfile POST	{contact_name,phone,company}; {saved:true}	Own profile only; no permission fields
submitPortalCost POST	{access_id,campaign,channel,spent_on,amount}; {saved:true}	Enabled member; insert cost; all spend triggers apply
submitPortalAdTotals POST	{access_id,campaign,channel,logged_on,spend,leads}; {saved:true}	Locked membership; Google/Meta daily replacement
savePortalBudget POST	{access_id,channel,amount:null or number}; {saved:true}	Enabled member, deal lock, selected platform cap only
submitPortalEarning POST	{access_id,customer_id,amount,earned_on,due_on}; {saved:true}	Service-only scoped earning insertion, unique lead
submitPortalPayment POST	{access_id,earning_id,submission_id,amount,paid_on,utr}; {saved:true}	Membership + earning lock, cap/replay verification

Operation	Input / return	Authority and persistence
savePartnerOffer POST	{deal_id,name,website_url,app_url,campaign_id,campaign_name,channel}; {id}	Owner saved LSP deal; strict schema
managePartnerOffer POST	{id,action:key/revoke/enable/disable}; {token or null}	Owner offer; hashed key/state mutation
submit_owner_payment SQL	earning ID, submission UUID, amount, paid date, UTR	Current owner from auth; earning lock and retry matching

Portal campaign names are trimmed 1–80; channels use the recorded nine-channel enum, while daily ad totals and platform budgets accept GOOGLE_ADS/META_ADS only. Portal cost >0 and ≤₹1 billion; portal daily spend may be zero. Platform budget nullable, finite 0–₹1 billion. Portal earnings positive, earned date ≤today, due ≥earned. Linked payments positive, earned date ≤paid date ≤today, UTR trimmed ≤60.

Invite claim is a conditional update on enabled/unexpired/unclaimed membership. Intake/portal lead RPCs lock their authorization rows and have a unique owner/submission index; concurrent reuse across different links can still hit unique-conflict rather than a friendly duplicate response. Billing extraction and composite dashboard reads are multi-request operations, not one database snapshot. Preserve this distinction; proposed hardening should use transactional/matching replay behavior where required.

19. Supplemental feature details

19.1 Customer planning exact derived states

Age = elapsed UTC days since shared_at, clamped at zero; stage age uses the most advanced recorded stage. Missing profile data includes bureau, salary, worst DPD, last-active days or state. Readiness precedence: disbursed → CONVERTED; approved → APPROVED; policy failure or NOT_ELIGIBLE segment → BLOCKED; missing data or NEEDS_REVIEW segment → REVIEW; otherwise PROMISING. Opportunity = approved or promising, not predicted approval probability.

Priority: converted→CLOSED; blocked→LOW; approved opportunity or opportunity aged ≥14 days→HIGH; promising or stage age ≥7 days→MEDIUM; otherwise LOW. Filters: age 0–7,8–30,31–60,>60; readiness all/opportunities/promising/approved/review/blocked/converted; priority high/medium/low/closed; sort priority/time-in-stage, oldest or newest. Last week is previous Monday-to-Monday; last month is previous calendar month. Inline stage date edits write on blur and are still subject to the database date-order rules. Delete removes a customer via the supplied owner/sample callback; backend must not bypass payout-linked integrity.

19.2 Campaign tracking detail

Control input spans 0–30%, disabled for Social; under 5% gets an experiment warning. Saving a valid plan selects it in Running. Tracking “Record day N results” generates synthetic deterministic daily metrics from assumed rates, not user-entered provider delivery evidence. Tracking charts show clicks, leads, cumulative disbursals and treated/control lift. Analysis allows owner to link/unlink saved customer campaign_id; this linkage supplies only recorded approval/disbursement outcomes, while lead/message tracking may still be simulated.

19.3 Payment management detail

Billing exposes generic contract transfer history with delete confirmation; Settle prefills outstanding. Earned tracker chooses an assigned lead with no earning, can prefill pipeline amount/dates and prohibits UI deletion when linked receipts exist. Detailed earning rows show paid/partly paid/unpaid. Owner linked-payment form chooses an unpaid earning and prefills remaining. Both receipt paths retain submission ID across retry but reset it after relevant form changes/success. Generic transfer notes ≤500; UTR ≤60.

19.4 Partner cost ratios versus owner cost ratios

Owner campaign CPL is zero when spend is zero but leads are positive. Partner campaign CPL/CPA/CPD are null unless both recorded spend and denominator are positive. Do not normalize these different current missing-spend semantics accidentally. Portal platform utilization is capped at 100%; at zero limit, any spend means 100%, otherwise zero. With null limit, remaining/utilization are unknown and overage is zero.

19.5 Offer form and reporting

Offer form selects a saved LSP deal and an existing browser-local campaign, then offering name and URLs. Cards show active/paused state, click count, confirmed downloads and recorded leads; tracking polls every 15 seconds. Keys are shown/copyable only immediately after generation; replace/revoke invalidates the previous key. Offers are unavailable in sample mode. Backend migration of local plans must retain campaign IDs used by these existing offers and customer links.

20. Application data dictionary

Source-derived table definitions from all 28 migrations. Initial fields are listed per table; later additions and constraints follow. Owner identifiers are UUIDs. Read all subsequent amendments before recreating a table. This appendix intentionally preserves SQL-level names for backend implementation.

billing_documents

Created in 0000_billing_documents.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
kind	text NOT NULL DEFAULT 'invoice'
file_path	text NOT NULL
file_name	text NOT NULL
mime_type	text NOT NULL
status	text NOT NULL DEFAULT 'extracting'
error	text
partner	text
doc_number	text
invoice_ref	text
doc_date	date
due_date	date
amount	numeric
gst	numeric
total	numeric
utr	text
created_at	timestamptz NOT NULL DEFAULT now()

customer_applications

Created in 0001_customer_applications.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
customer_name	text NOT NULL
mobile	text
partner_code	text NOT NULL
state	text
bureau_score	integer
monthly_salary	numeric
worst_dpd_12m	integer
last_active_days	integer
shared_at	date NOT NULL DEFAULT current_date
policy_passed	boolean NOT NULL DEFAULT true
reject_reason	text
applied_at	date
approved_at	date
approved_amount	numeric
disbursed_at	date
disbursed_amount	numeric
created_at	timestamptz NOT NULL DEFAULT now()

partner_deals

Created in 0002_create_partner_deals.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
code	text NOT NULL
name	text NOT NULL
partner_type	text NOT NULL DEFAULT 'LSP'
model	text NOT NULL CHECK (model IN ('CPL','CPA','CPD'))
kind	text NOT NULL CHECK (kind IN ('FLAT','PERCENTAGE'))
rate	numeric NOT NULL CHECK (rate >= 0 AND rate <> 'NaN'::numeric AND rate <> 'Infinity'::numeric)
min_payout	numeric CHECK (min_payout >= 0 AND min_payout <> 'NaN'::numeric AND min_payout <> 'Infinity'::numeric)
max_payout	numeric CHECK (max_payout >= 0 AND max_payout <> 'NaN'::numeric AND max_payout <> 'Infinity'::numeric)
gst_inclusive	boolean NOT NULL DEFAULT false
starts_at	date NOT NULL DEFAULT CURRENT_DATE
ends_at	date
payment_days	integer NOT NULL DEFAULT 30 CHECK (payment_days >= 0)
notes	text NOT NULL DEFAULT ''
created_at	timestamptz NOT NULL DEFAULT now()
UNIQUE	(user_id,code)
CHECK	(length(trim(code)) > 0 AND code = upper(trim(code)))
CHECK	(length(trim(name)) > 0)
CHECK	(model <> 'CPL' OR kind = 'FLAT')
CHECK	(kind <> 'PERCENTAGE' OR rate <= 100)
CHECK	(min_payout IS NULL OR max_payout IS NULL OR min_payout <= max_payout)
CHECK	(ends_at IS NULL OR ends_at >= starts_at)

business_reminders

Created in 0004_overview_reminders.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
title	text NOT NULL CHECK (length(trim(title)) BETWEEN 1 AND 200)
due_at	timestamptz NOT NULL
priority	text NOT NULL DEFAULT 'NORMAL' CHECK (priority IN ('HIGH','NORMAL','LOW'))
completed_at	timestamptz
created_at	timestamptz NOT NULL DEFAULT now()

partner_intake_links

Created in 0006_partner_intake_links.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
deal_id	uuid NOT NULL UNIQUE REFERENCES public.partner_deals(id) ON DELETE CASCADE
token_hash	text NOT NULL UNIQUE CHECK (token_hash ~ '^[a-f0-9]{64}$')
enabled	boolean NOT NULL DEFAULT true
created_at	timestamptz NOT NULL DEFAULT now()

profiles

Created in 0007_partner_portal_access.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY REFERENCES auth.users(id) ON DELETE CASCADE
contact_name	text NOT NULL DEFAULT ''
phone	text NOT NULL DEFAULT ''
company	text NOT NULL DEFAULT ''
CONSTRAINT	profile_contact_lengths CHECK (length(contact_name)<=200 AND length(phone)<=30 AND length(company)<=200)

partner_portal_access

Created in 0007_partner_portal_access.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
owner_id	uuid NOT NULL DEFAULT auth.uid()
deal_id	uuid NOT NULL REFERENCES public.partner_deals(id) ON DELETE CASCADE
invited_email	text NOT NULL
token_hash	text NOT NULL UNIQUE
member_id	uuid
enabled	boolean NOT NULL DEFAULT true
created_at	timestamptz NOT NULL DEFAULT now()
expires_at	timestamptz NOT NULL DEFAULT (now()+interval '7 days')
UNIQUE(deal_id,invited_email)	
CONSTRAINT	portal_email_length CHECK(length(invited_email) BETWEEN 3 AND 320 AND invited_email=lower(trim(invited_email)))

partner_payouts

Created in 0008_partner_payouts.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
deal_id	uuid NOT NULL REFERENCES public.partner_deals(id) ON DELETE CASCADE
amount	numeric NOT NULL CHECK (amount > 0)
paid_on	date NOT NULL DEFAULT CURRENT_DATE
utr	text NOT NULL DEFAULT '' CHECK (char_length(utr) <= 60)
notes	text NOT NULL DEFAULT '' CHECK (char_length(notes) <= 500)
created_at	timestamptz NOT NULL DEFAULT now()

partner_campaign_costs

Created in 0010_partner_campaign_sources.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
owner_id	uuid NOT NULL
deal_id	uuid NOT NULL REFERENCES public.partner_deals(id) ON DELETE CASCADE
campaign	text NOT NULL CHECK (char_length(trim(campaign)) BETWEEN 1 AND 80)
channel	text NOT NULL CHECK (channel IN ('SMS','WHATSAPP','EMAIL','SOCIAL','CALL','WALKIN','OTHER'))
spent_on	date NOT NULL CHECK (spent_on <= CURRENT_DATE + 1)
amount	numeric NOT NULL CHECK (amount > 0 AND amount <= 1000000000)
created_by	uuid NOT NULL
created_at	timestamptz NOT NULL DEFAULT now()

spend_entries

Created in 0011_own_spend_entries.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
campaign_id	text CHECK (campaign_id IS NULL OR char_length(campaign_id) <= 60)
channel	text NOT NULL CHECK (channel IN ('SMS','WHATSAPP','META'))
spent_on	date NOT NULL CHECK (spent_on <= current_date + 1)
amount	numeric NOT NULL CHECK (amount > 0 AND amount <= 1000000000)
notes	text CHECK (notes IS NULL OR char_length(notes) <= 300)
created_at	timestamptz NOT NULL DEFAULT now()

partner_earned_payouts

Created in 0013_partner_earned_payouts.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
deal_id	uuid NOT NULL REFERENCES public.partner_deals(id)
customer_id	uuid NOT NULL REFERENCES public.customer_applications(id)
amount	numeric NOT NULL CHECK (amount > 0)
earned_on	date NOT NULL
due_on	date NOT NULL CHECK (due_on >= earned_on)
created_at	timestamptz NOT NULL DEFAULT now()
UNIQUE	(user_id, customer_id)

partner_offers

Created in 0016_partner_offer_tracking.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
deal_id	uuid NOT NULL REFERENCES public.partner_deals(id) ON DELETE CASCADE
name	text NOT NULL CHECK(length(name) BETWEEN 1 AND 120)
website_url	text NOT NULL CHECK(website_url ~ '^https://' AND length(website_url)<=2048)
app_url	text CHECK(app_url IS NULL OR (app_url ~ '^https://' AND length(app_url)<=2048))
campaign_id	text NOT NULL CHECK(length(campaign_id) BETWEEN 1 AND 60)
campaign_name	text NOT NULL CHECK(length(campaign_name) BETWEEN 1 AND 80)
channel	text NOT NULL CHECK(channel IN ('SMS','WHATSAPP','EMAIL','SOCIAL','GOOGLE_ADS','META_ADS','CALL','WALKIN','OTHER'))
enabled	boolean NOT NULL DEFAULT true
api_key_hash	text CHECK(api_key_hash ~ '^[a-f0-9]{64}$')
created_at	timestamptz NOT NULL DEFAULT now()

partner_offer_clicks

Created in 0016_partner_offer_tracking.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
offer_id	uuid NOT NULL REFERENCES public.partner_offers(id) ON DELETE CASCADE
user_id	uuid NOT NULL
target	text NOT NULL CHECK(target IN ('website','app'))
created_at	timestamptz NOT NULL DEFAULT now()

partner_offer_events

Created in 0016_partner_offer_tracking.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
offer_id	uuid NOT NULL REFERENCES public.partner_offers(id) ON DELETE CASCADE
click_id	uuid NOT NULL REFERENCES public.partner_offer_clicks(id) ON DELETE CASCADE
user_id	uuid NOT NULL
event_id	uuid NOT NULL
type	text NOT NULL CHECK(type IN ('DOWNLOAD','LEAD'))
customer_id	uuid REFERENCES public.customer_applications(id) ON DELETE SET NULL
created_at	timestamptz NOT NULL DEFAULT now()
UNIQUE(offer_id,event_id)	
UNIQUE(click_id,type)	

owner_ad_campaign_logs

Created in 0018_owner_ad_campaign_logs.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
platform	text NOT NULL CHECK (platform IN ('google','meta'))
campaign	text NOT NULL CHECK (length(btrim(campaign)) BETWEEN 1 AND 80)
logged_on	date NOT NULL
spend	numeric(14,2) NOT NULL CHECK (spend >= 0)
leads	integer NOT NULL CHECK (leads >= 0)
created_at	timestamptz NOT NULL DEFAULT now()
UNIQUE(user_id,platform,campaign,logged_on)	

campaign_budgets

Created in 0020_campaign_budgets.sql

Field / constraint	Type, default and validation
id	uuid PRIMARY KEY DEFAULT gen_random_uuid()
user_id	uuid NOT NULL DEFAULT auth.uid()
platform	text NOT NULL CHECK (platform IN ('google','meta'))
campaign	text NOT NULL CHECK (char_length(btrim(campaign)) BETWEEN 1 AND 80 AND campaign=btrim(campaign))
amount	numeric(14,2) NOT NULL CHECK (amount > 0 AND amount <= 999999999999.99)
UNIQUE(user_id,platform,campaign)	

21. Schema amendments and database routines

Later migration amendments below extend the original table definitions. Constraints, indexes, grants, triggers and SQL function bodies remain in the ordered migration source; do not replace them with UI-only validation.

0003_partner_policy_limits.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD COLUMN bureau_min integer, ADD COLUMN bureau_max integer, ADD COLUMN dpd_max integer;
```

0003_partner_policy_limits.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD CONSTRAINT partner_policy_limits_valid CHECK ((bureau_min IS NULL OR bureau_min BETWEEN 0 AND 900) AND (bureau_max IS NULL OR bureau_max BETWEEN 0 AND 900) AND (bureau_min IS NULL OR bureau_max IS NULL OR bureau_min <= bureau_max) AND (dpd_max IS NULL OR dpd_max >= 0));
```

0006_partner_intake_links.sql · customer_applications

```
ALTER TABLE public.customer_applications ADD COLUMN lead_source text NOT NULL DEFAULT 'manual', ADD COLUMN intake_submission_id uuid;
```

0009_customer_campaign_attribution.sql · customer_applications

```
ALTER TABLE public.customer_applications ADD COLUMN campaign_id text CHECK (campaign_id IS NULL OR char_length(campaign_id) <= 60);
```

0010_partner_campaign_sources.sql · customer_applications

```
ALTER TABLE public.customer_applications
ADD COLUMN lead_channel text CHECK (lead_channel IN
('SMS','WHATSAPP','EMAIL','SOCIAL','CALL','WALKIN','OTHER')),
ADD COLUMN partner_campaign text CHECK (char_length(partner_campaign) BETWEEN 1 AND 80);
```

0012_partner_ad_channels.sql · customer_applications

```
ALTER TABLE public.customer_applications ADD CONSTRAINT customer_applications_lead_channel_check CHECK (lead_channel IN ('SMS','WHATSAPP','EMAIL','SOCIAL','GOOGLE_ADS','META_ADS','CALL','WALKIN','OTHER'));
```

0012_partner_ad_channels.sql · partner_campaign_costs

```
ALTER TABLE public.partner_campaign_costs ADD CONSTRAINT partner_campaign_costs_channel_check CHECK (channel IN ('SMS','WHATSAPP','EMAIL','SOCIAL','GOOGLE_ADS','META_ADS','CALL','WALKIN','OTHER'));
```

0014_portal_earned_payout_capture.sql · partner_earned_payouts

```
ALTER TABLE public.partner_earned_payouts ADD COLUMN created_by uuid;
```

0015_linked_portal_payments.sql · partner_payouts

```
ALTER TABLE public.partner_payouts ADD COLUMN earned_payout_id uuid REFERENCES public.partner_earned_payouts(id) ON DELETE RESTRICT;
```

0015_linked_portal_payments.sql · partner_payouts

```
ALTER TABLE public.partner_payouts ADD COLUMN created_by uuid;
```

0015_linked_portal_payments.sql · partner_payouts

```
ALTER TABLE public.partner_payouts ADD COLUMN submission_id uuid;
```

0016_partner_offer_tracking.sql · customer_applications

```
ALTER TABLE public.customer_applications ADD COLUMN partner_offer_id uuid REFERENCES
public.partner_offers(id) ON DELETE SET NULL;
```

0019_partner_daily_ad_totals.sql · partner_campaign_costs

```
ALTER TABLE public.partner_campaign_costs ADD COLUMN reported_leads integer;
```

0019_partner_daily_ad_totals.sql · partner_campaign_costs

```
ALTER TABLE public.partner_campaign_costs ADD CONSTRAINT partner_reported_leads_nonnegative CHECK
(reported_leads IS NULL OR reported_leads >= 0);
```

0021_owner_campaign_revenue.sql · owner_ad_campaign_logs

```
ALTER TABLE public.owner_ad_campaign_logs ADD COLUMN revenue numeric(14,2) CHECK (revenue >= 0 AND revenue
<= 999999999999.99);
```

0022_partner_campaign_spend_limits.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD COLUMN campaign_budget numeric(14,2), ADD COLUMN campaign_spend_limit
numeric(14,2);
```

0022_partner_campaign_spend_limits.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD CONSTRAINT partner_campaign_budget_valid CHECK (campaign_budget IS
NULL OR (campaign_budget>=0 AND campaign_budget<=1000000000)), ADD CONSTRAINT partner_campaign_limit_valid
CHECK (campaign_spend_limit IS NULL OR (campaign_spend_limit>=0 AND campaign_spend_limit<=1000000000)), ADD
CONSTRAINT partner_campaign_budget_below_limit CHECK (campaign_budget IS NULL OR campaign_spend_limit IS
NULL OR campaign_budget<=campaign_spend_limit);
```

0024_partner_portal_platform_budgets.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD COLUMN partner_google_spend_limit numeric(14,2), ADD COLUMN
partner_meta_spend_limit numeric(14,2);
```

0024_partner_portal_platform_budgets.sql · partner_deals

```
ALTER TABLE public.partner_deals ADD CONSTRAINT partner_platform_limits_valid CHECK
((partner_google_spend_limit IS NULL OR (partner_google_spend_limit >= 0 AND partner_google_spend_limit <=
1000000000)) AND (partner_meta_spend_limit IS NULL OR (partner_meta_spend_limit >= 0 AND
partner_meta_spend_limit <= 1000000000)));
```

0027_payout_recording_source.sql · partner_earned_payouts

```
ALTER TABLE public.partner_earned_payouts ADD COLUMN recorded_via text;
```

0027_payout_recording_source.sql · partner_payouts

```
ALTER TABLE public.partner_payouts ADD COLUMN recorded_via text;
```

Routine inventory by migration

Migration	Created/replaced database routines
0001_customer_applications.sql	validate_customer_application
0006_partner_intake_links.sql	submit_partner_intake
0007_partner_portal_access.sql	create_portal_profile, submit_portal_lead
0010_partner_campaign_sources.sql	submit_portal_lead
0012_partner_ad_channels.sql	submit_portal_lead
0013_partner_earned_payouts.sql	validate_partner_earning_date
0014_portal_earned_payout_capture.sql	submit_portal_earning
0015_linked_portal_payments.sql	submit_portal_payment
0016_partner_offer_tracking.sql	record_partner_offer_click, record_partner_offer_event
0017_owner_linked_partner_payments.sql	submit_owner_payment
0018_owner_ad_campaign_logs.sql	validate_owner_ad_log_date
0019_partner_daily_ad_totals.sql	submit_portal_ad_totals
0022_partner_campaign_spend_limits.sql	enforce_partner_campaign_spend_limit
0023_partner_limit_daily_replacement.sql	enforce_partner_campaign_spend_limit
0024_partner_portal_platform_budgets.sql	save_partner_platform_budget, enforce_partner_platform_budget
0025_partner_portal_remove_budget.sql	remove_partner_platform_budget
0026_protect_lead_partner_assignment.sql	protect_lead_partner_assignment, lock_earned_lead_partner
0027_payout_recording_source.sql	set_payout_recording_source

22. Source inventory and handoff notes

```
Content route: src/routes/ads.tsx
Content route: src/routes/advisor.tsx
Content route: src/routes/billing.tsx
Content route: src/routes/campaigns.tsx
Content route: src/routes/criteria.tsx
Content route: src/routes/customers.tsx
Content route: src/routes/economics.tsx
Content route: src/routes/funnel.tsx
Content route: src/routes/index.tsx
Content route: src/routes/insights.tsx
Content route: src/routes/intake.$token.tsx
Content route: src/routes/partner-deals.tsx
Content route: src/routes/partner-onboarding.tsx
Content route: src/routes/partner-performance.$code.tsx
Content route: src/routes/partner-policies.tsx
Content route: src/routes/partners.tsx
Content route: src/routes/payments.tsx
Content route: src/routes/portal.tsx
Content route: src/routes/today.tsx
HTTP handler: src/routes/api/advisor.ts
HTTP handler: src/routes/api/public/partner-events.ts
HTTP handler: src/routes/api/public/partner-offers.ts
HTTP handler: src/routes/api/public/partner-redirect.ts
```

Primary source areas

src/components/dash: feature screens and controls. src/lib: validation, pure business rules, providers, owner hooks and server operations. drizzle/migrations/0000–0027: ordered application schema and database enforcement. AGENTS.md: architecture rules. Project memory: owner business/design constraints.

Documentation verification

This document was assembled from route/component source, server schemas/functions, ordered migrations, pure calculations and tests. It does not reproduce credentials, private live customer data or secret keys. It is not a live schema dump and does not confirm hosted auth/provider settings. Some older roadmap verification notes are historical; source behavior takes precedence over stale blockers.

Suggested handoff use

Use chapters 1–11 for tab-wise functional scope; 12–16 for architecture, rules, security, delivery and limitations; 17–19 for contracts and edge cases; 20–22 for data dictionary, migration/routine inventory and source mapping. Confirm business changes before replacing provisional models or reconciliation semantics.