Sentinel Support: Full-Stack Fintech Case Resolution with Multi-Agent Automation

Build a production-minded **full-stack** system for internal support agents to (1) ingest and explore transactions, (2) generate **AI insights & reports**, and (3) **auto-resolve cases** (freeze card, open dispute, contact customer) via a **multi-agent** pipeline. The app must be **local/offline runnable**, have **deterministic fallbacks**, and expose **traces, metrics, and audit** artifacts.

Stack (required):

- **Frontend:** React + TypeScript (Vite or Next.js CSR mode), Tailwind (or CSS Modules)
- **Backend:** Node.js (TypeScript) + Express (or Fastify)
- **DB: PostgreSQL** (via Prisma/TypeORM/SQL)
- **Cache/Queue:** Redis (rate-limit + background jobs)
- **Infra:** Docker Compose (pg + redis + api + web)
- **Optional LLM:** behind a flag with deterministic fallback (no network required to pass)

1) What you're building (one sentence)

A **case-resolution console** where a support agent can load customer activity, get **AI-generated insights**, and run a **multi-agent triage** that recommends and executes safe actions, with **explainable traces**, **policy guardrails**, and **observability**.

2) Core Capabilities (must ship)

A) Frontend (React + TS)

- **Routes**
  - `/dashboard`: KPIs (alerts in queue, disputes opened, avg triage latency), quick filters.
  - `/alerts`: paginated/virtualized queue with risk score + "Open Triage".
  - `/customer/:id`: transactions timeline, category spend, merchant mix, anomalies.
  - `/evals`: run & view eval results (pass/fail, confusion matrix, top failures).
- **Triage Drawer (the hero)**
  - Shows **risk score**, **top reasons**, **plan**, **tool calls** (ok/error/duration), **fallbacks**, **citations**, **recommended action**.
  - Buttons: **Freeze Card**, **Open Dispute**, **Contact Customer**, **Mark False Positive**.
  - **Streaming updates** (SSE or WebSocket); drawer is fully **keyboard accessible**.

- **A11y & Perf**
    - Focus trap, ESC to close, return focus; ARIA for dialog; polite live region for streamed updates.
    - Virtualized tables for ≥2k rows; memoized rows; no jank.

B) Backend (Node + Express/Fastify + TS)

- **APIs**
    - `POST /api/ingest/transactions` (CSV or JSON) → upsert, dedupe by `(customerId, txnId)`.
    - `GET /api/customer/:id/transactions?from=&to=&cursor=&limit=` → **keyset pagination**.
    - `GET /api/insights/:customerId/summary` → categories, merchants, anomalies, monthly trend.
    - `POST /api/triage` → starts a triage run (`runId`), streams events via **SSE** `GET /api/triage/:runId/stream`.
    - `POST /api/action/freeze-card` (API key + optional OTP) → `PENDING_OTP|FROZEN`.
    - `POST /api/action/open-dispute` → `{ caseId, status:"OPEN" }`.
    - `GET /api/kb/search?q=` → `{ results:[{docId,title,anchor,extract}] }`.
    - `GET /metrics`, `GET /health`.
- **Cross-cutting**
    - **Rate-limit** 5 r/s per client (token bucket in Redis) → 429 with `Retry-After`.
    - **Idempotency-Key** on ingest & actions; return prior result on replay.
    - **Observability**: Prometheus metrics + structured JSON logs.
    - **Audit**: every action appends case events (who/what/when, redacted payload).

C) Multi-Agent Orchestration (server-side)

- **Orchestrator (Planner):** builds a bounded plan and executes sub-agents with timeouts & retries.
    - Default plan: `["getProfile","recentTx","riskSignals","kbLookup","decide","proposeAction"]`
- **Sub-Agents** (tool-using)
    - **Insights Agent:** categories, merchant concentration, spend patterns (deterministic rules; optional LLM phrasing).
    - **Fraud Agent:** velocity, device change, MCC rarity, prior chargebacks → `{score, reasons[], action?}`.
    - **KB Agent:** retrieve cited answers from local JSON (title + anchor).
    - **Compliance Agent:** OTP/identity gate, policy deny (e.g., unfreeze without verification).

    - **Redactor:** PAN-like 13–19 digits → `****REDACTED****`; mask emails; scrub logs & traces.
    - **Summarizer:** customer message & internal note (template fallback).
- **Guardrails**
    - Tool **timeouts ≤1s**; flow budget **≤5s**.
    - **Retries**: max 2 (150ms, 400ms + jitter).
    - **Circuit breaker**: open 30s after 3 consecutive failures per tool.
    - **Schema validation**: Zod/JSON-Schema for tool I/O (reject/annotate trace on mismatch).
    - **Prompt-injection**: user text cannot trigger tools without policy check; sanitize inputs.

3) Data Model (PostgreSQL)

```
-- customers/cards/accountscustomers(id pk, name, email_masked, kyc_level,
created_at)
cards(id pk, customer_id fk, last4, network, status, created_at)
accounts(id pk, customer_id fk, balance_cents, currency)
-- transactionstransactions(
  id pk, customer_id fk, card_id fk,
  mcc, merchant, amount_cents, currency,
  ts timestamptz, device_id, country, city
);-- indexes: (customer_id, ts DESC), (merchant), (mcc), (customer_id,
merchant)-- alerts & casesalerts(id pk, customer_id fk, suspect_txn_id fk,
created_at, risk text, status text)
cases(id pk, customer_id fk, txn_id fk null, type text, status text,
reason_code text, created_at)
case_events(id pk, case_id fk, ts, actor, action, payload_json)
-- triage runs & tracestriage_runs(id pk, alert_id fk, started_at, ended_at,
risk text, reasons jsonb, fallback_used bool, latency_ms int)
agent_traces(run_id fk, seq int, step text, ok bool, duration_ms int,
detail_json jsonb, primary key (run_id, seq))
-- kb & policieskb_docs(id pk, title, anchor, content_text)
policies(id pk, code, title, content_text)
```

4) Fixtures (must include)

```
/fixtures
  customers.json   cards.json   accounts.json
  transactions.json (≥ 200k rows; provide a generator script to reach ≥ 1M
locally)
  alerts.json      kb_docs.json policies.json  chargebacks.json  devices.json
  evals/*.json     (acceptance/golden cases; see §7)
```

5) Hard Requirements (SLOs/Safety)

- **Local run**: `docker compose up` brings everything online; seed scripts included.
- **Performance**: `/customer/:id/transactions?last=90d` **p95 ≤ 100ms** on a dataset ≥ 1,000,000 rows.
- **Streaming**: triage events stream to the FE (SSE or WS); resilient to reconnects.
- **Security**:
  - PAN-like sequences **never** appear in UI/logs/traces (13–19 digits redacted).
  - Mutations require `X-API-Key`; RBAC: "agent" vs "lead" (lead can force-approve); audit all actions.
  - **CSP** suitable for sensitive data pages (no `unsafe-inline`).
- **Correctness**: idempotent actions; 429 honored; evals pass (see §7).

6) APIs (contracts)

- `POST /api/ingest/transactions → { accepted: true, count, requestId }`
- `GET /api/customer/:id/transactions?from=&to=&cursor=&limit=`
  - **Keyset** pagination: returns `{ items:[], nextCursor }`
- `GET /api/insights/:customerId/summary`
- `POST /api/triage → { runId, alertId }`
- `GET /api/triage/:runId/stream` (SSE) → events: `plan_built`, `tool_update`, `fallback_triggered`, `decision_finalized`
- `POST /api/action/freeze-card` (Idempotency-Key, API key)
  - Req: `{ cardId, otp? } → { status:"PENDING_OTP"|"FROZEN", requestId }`
- `POST /api/action/open-dispute`
  - Req: `{ txnId, reasonCode, confirm } → { caseId, status:"OPEN" }`
- `GET /api/kb/search?q=`
- `GET /metrics`, `GET /health`

7) Acceptance Scenarios (we will verify exactly)

1. **Freeze w/ OTP path**
   a. From `/alerts`, open triage on an alert.
   b. **Expected:** recommendation = "Freeze Card", **OTP required**; enter `123456` → status `FROZEN`; trace shows `freezeCard ok`; metrics increment `action_blocked_total{policy=otp_required}`.
2. **Dispute creation**

   a. Unrecognized ₹4,999 at "ABC Mart" yesterday.
   b. **Expected:** single txn match; reason **10.4** proposed; **case OPEN** with `caseId` in timeline; citation to KB **Disputes**.
3. **Duplicate pending vs captured**
   a. "Charged twice at QuickCab."
   b. **Expected:** explanation (preauth vs capture); **no dispute**; risk = `low|medium`; KB citation appears.
4. **Risk tool timeout → fallback**
   a. Simulate `riskSignals` failure.
   b. **Expected:** `fallback_used=true`, risk ≤ `medium`, reason includes `risk_unavailable`; SSE shows `fallback_triggered`.
5. **429 behavior**
   a. Trigger rate limit (spam triage).
   b. **Expected:** 429 with `Retry-After`; FE disables control ~given ms; **no duplicate triage runs**; next attempt succeeds.
6. **PII redaction**
   a. User input or tool detail contains `4111111111111111`.
   b. **Expected:** redacted as `****REDACTED****` in UI/logs/trace; structured log line includes `masked=true`.
7. **Performance**
   a. With ≥1M txns, `/customer/:id/transactions?last=90d` **p95 ≤ 100ms**; show timing + `EXPLAIN ANALYZE` snippet in README.

8) Metrics & Logs (what we expect to see non-zero)

- **Metrics**
  - `api_request_latency_ms` (hist)
  - `agent_latency_ms` (hist)
  - `tool_call_total{tool,ok}`
  - `agent_fallback_total{tool}`
  - `rate_limit_block_total`
  - `action_blocked_total{policy}`
- **Structured logs (JSON)**
  - fields: `ts, level, requestId, runId, sessionId, customerId_masked, event, masked=true`
  - key events: `plan_built`, `tool_invoked`, `fallback_triggered`, `decision_finalized`, `action_completed`

9) Evals (golden set & CLI)

- Provide **≥12 eval cases** in `/fixtures/evals/*.json`—cover all acceptance scenarios + ambiguous merchants + device change + travel window cases.
- CLI `npm run eval` prints:
  - **Task success rate**, **fallback rate** by tool
  - **Agent latency p50/p95**
  - **Risk confusion matrix** (`low|medium|high`)
  - **Top policy denials**

10) Deliverables

- Monorepo (or two folders): `/web` (React), `/api` (Node), `/fixtures`, `/scripts`, `/docs`.
- **Docker Compose** for pg, redis, api, web.
- **README (≤1 page)**: run in ≤3 commands, arch diagram (ASCII ok), key trade-offs.
- **ADR.md**: 8–12 bullets (why keyset pagination, why SSE, why schema X, etc.).
- **Postman/HTTP** collection.
- **Demo video (≤8 min)**: show freeze (OTP), dispute, fallback, 429, metrics/logs.
- **Eval report** (from CLI output) + brief commentary.

```
{ "topMerchants":[{"merchant":"ABC","count":12}],"categories":[{"name":"Transport","pct":0.23}],"monthlyTrend":[{"month":"2025-07","sum":120045}],"anomalies":[{"ts":"2025-07-13","z":3.1,"note":"spike"}] }
```