TECHNICAL SPECIFICATION · SECURITY & COMPLIANCE

Enterprise Loyalty Management System

A configuration-driven, ledger-safe loyalty platform —
Admin Console + Member Portal, engineered for financial-grade integrity.

VERSION
1.0 — Technical Baseline

DEPLOYMENT MODEL
Single-tenant, custom-built

PLATFORM TYPE
Web (Admin Console + Member Portal)

JURISDICTION
India — DPDP Act 2023

CORE STACK
React + TypeScript · FastAPI · MongoDB

CLIENT PROJECT OWNER
Vineet Narang

PREPARED FOR (CLIENT)

BharatPe

Powered by **Fundle** · www.fundle.ai

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

Table of Contents

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

21 Testing Strategy

0. Document Control & Reading Guide

Audience: Solution architects, backend/frontend engineers, security/compliance reviewers, client project owner (Vineet Narang).

Stack (hardened, non-negotiable for this build): React + TypeScript (front end), FastAPI (Python) services, MongoDB (primary datastore).

> DEVIATION FROM SOURCE BRIEF The Master Brief suggested PostgreSQL/Supabase. This spec uses **MongoDB** with multi-document ACID transactions + an append-only, hash-chained ledger to satisfy financial-grade integrity. Trade-offs documented in §7.

Design philosophy: Configuration over code; engine separation; append-only ledger; idempotent events; maker-checker governance; audit-everything.

> PREPARED BY · DOCUMENT CONTACTS Prepared by **Fundle** (www.fundle.ai) for the client **BharatPe**. Document contacts: Anmol Berry (Anmol.berry@fundle.ai) · Deepak Tyagi (Deepak.tyagi@fundle.ai).

1. Scope & Objectives

In scope (production system)

- Single enterprise tenant with internal hierarchy: `Enterprise → BU → SBU → Stakeholder Segment (SHS) → Tier/Segment → Geography/Territory → Program/Scheme → Member`.
- Primary worked domain: **B2B / channel loyalty** (dealers, distributors, retailers). Secondary: **B2C member** loyalty.
- Core engines: Identity, Event Ingestion, Rules, Ledger, Tier, Rewards/Redemption, Campaign, Gamification, Journey/Engagement, Analytics, Integration, Workflow/Approvals, Fraud/Reconciliation.
- Admin Console (desktop-first) + Member Portal (mobile-responsive web).

Out of scope (v1)

- Native mobile apps (member portal is responsive web).
- Direct money movement / card issuance (treated as integration boundary; requires separate licensing decision — see §18.5).
- Real ML models for personalization (illustrative analytics with real drill-down; §13).

> SUCCESS CRITERION **Configurability proven end-to-end:** a config change in Admin (e.g., a campaign) demonstrably flows through event → rule evaluation → ledger → member balance/tier → analytics.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

2. High-Level Architecture

2.1 Style

Modular service-oriented monorepo. Logical service boundaries enforced even if deployed as a modular monolith initially (single-tenant does not need microservice sprawl on day one).

2.2 Logical services (FastAPI modules/routers)

Service	Responsibility
`identity-svc`	Auth, users, RBAC, SSO/MFA, member identity, KYC state
`masterdata-svc`	Org hierarchy, geo, product/SKU, tier defs, vendor, catalogue, tax, periods
`event-svc`	Event ingestion, validation, idempotency, staging queue
`rules-svc`	Rule builder, versioning, simulation, evaluation, conflict resolution
`ledger-svc`	Append-only points/cashback ledger, balances, expiry, liability
`tier-svc`	Tier qualification, upgrade/downgrade, grace, benefits
`rewards-svc`	Catalogue, redemption, orders, vendor fulfilment, auto-refund
`campaign-svc`	Campaign lifecycle, audience, budgets, approvals
`gamification-svc`	Missions, badges, streaks, leaderboards
`journey-svc`	Segmentation, journey orchestration, notification dispatch
`analytics-svc`	Dashboards, RFM/CLTV, liability, campaign ROI, exports
`integration-svc`	Adapter framework, webhooks, SFTP/batch, retry/DLQ, integration logs
`workflow-svc`	Maker-checker, approval matrices, SLA/escalation
`fraud-svc`	Velocity rules, duplicate/QR abuse detection, reconciliation
`audit-svc`	Immutable audit log, evidence retrieval

2.3 Communication

Synchronous REST (internal + external). Asynchronous processing via a message queue for event ingestion, ledger posting, notifications, and reconciliation. Recommended: **Redis Streams** (lightweight, single-tenant scale) or **RabbitMQ** if ordering/DLQ guarantees needed. Kafka is over-engineering for single-tenant v1 — flag if volumes justify later.

2.4 Async pattern

Event → staging → rule evaluation → ledger posting are decoupled worker stages with idempotency keys and correlation IDs threaded through.

2.5 Deployment topology (single-tenant)

- One isolated environment per client (dev/staging/prod).
- Containerized (Docker) services; orchestration via Kubernetes or a managed container platform. For a single tenant, a right-sized K8s namespace or Docker Compose + managed hosting is acceptable for v1 — do not over-provision.
- MongoDB replica set (3 nodes) for HA + transaction support (transactions require a replica set).
- Data residency: **all infrastructure and backups in an India region** (§18.4).

3. Technology Stack (Detailed)

Front end

- React 18 + TypeScript, Vite build.
- Component library with accessible primitives (Radix UI / shadcn) + Tailwind.
- State/data: TanStack Query (server cache) + lightweight client state (Zustand).
- Two apps in monorepo: `admin-console` (desktop-first, dense) and `member-portal` (mobile-responsive, warm).
- Charts: Recharts / visx for analytics drill-downs.

Back end

- FastAPI (Python 3.11+), Pydantic v2 for schema validation, async I/O.
- Auth: OAuth2/OIDC, JWT access tokens + refresh, MFA (TOTP), SSO (SAML/OIDC) for admin.
- Background workers: Celery / Arq with Redis broker.
- API docs: auto-generated OpenAPI (FastAPI native).

Data

- MongoDB (replica set) — primary store.
- Redis — cache, rate limiting, queues, session/token blacklist.
- Object storage (S3-compatible, India region) — documents, KYC artifacts, exports, catalogue media.

Observability

OpenTelemetry traces, Prometheus metrics, structured JSON logs (correlation IDs), Grafana/Loki dashboards, alerting.

4. Data Model (MongoDB)

4.1 Design principles

- Effective-dating and versioning on all configuration entities (rules, tiers, catalogue, masters): `version`, `effective_from`, `effective_to`, `status` (draft/published/retired).
- Soft-delete/inactivation, never hard-delete configuration or member records.
- Audit columns on every document: `created_at`, `created_by`, `updated_at`, `updated_by`.
- Data classification tag on PII fields (§18.2).

4.2 Core collections (representative schemas)

`organizations` — hierarchy nodes

```
{ _id, node_type: "BU|SBU|SHS|Territory|Program", parent_id, name, code,
  attributes: {}, status, effective_from, effective_to, audit }
```

`members` (covers B2B associations + B2C members)

```
{ _id, member_type: "dealer|distributor|retailer|consumer|employee",
  external_ref, name, contact: {phone, email}, org_scope: {bu, sbu, shs, territory},
  kyc: {status, level, verified_at, provider_ref}, consent: [{purpose, granted, ts}],
  tier: {current_tier_id, since, qualifying_period}, status, pii_class, audit }
```

`events` (immutable, append-only)

```
{ _id, event_id (unique), idempotency_key (unique), correlation_id, event_type,
  source: "POS|API|file|QR|manual", occurred_at, received_at, member_id,
  org_scope: {bu,sbu,shs,territory}, amount, quantity, currency: "INR",
  sku, category, location, device, payload, version, processing_status }
```

> **DEFENSE IN DEPTH** Unique index on `idempotency_key` enforces dedupe at the DB layer (beyond app checks).

`rules`

```
{ _id, rule_id, name, type: "earn|redeem|tier|campaign", version, status,
  dimensions: {org, member, geo, product, transaction, time, behavior},
  conditions: [], actions: [], priority, stacking_policy, exclusivity,
  effective_from, effective_to, simulation_meta, approval_ref, audit }
```

`ledger_entries` (append-only, hash-chained — see §7)

```
{ _id, entry_id, account_id (member/points account), entry_type: "accrual|burn|
  pending|reversal|expiry|adjustment|transfer", points, balance_after,
  currency: "INR", source_event_id, rule_version_id, campaign_id,
  reason, actor, prev_hash, entry_hash, posted_at, immutable: true }
```

`points_accounts`

```
{ _id, member_id, program_id, balance, pending_balance, lifetime_earned,
  lifetime_burned, liability_value, last_reconciled_at }
```

Other collections

tiers · tier_history · catalogue_items · redemptions · orders · campaigns · campaign_budgets · gamification_missions · member_progress · journeys · journey_runs · notifications · approvals · tickets · integration_logs · reconciliation_runs · audit_log · users · roles

4.3 Indexing strategy (critical for scale)

- `events`: unique(`idempotency_key`), unique(`event_id`), compound(`member_id`, `occurred_at`), (`processing_status`).
- `ledger_entries`: (`account_id`, `posted_at`), unique(`entry_id`), (`source_event_id`).
- `members`: (`external_ref`), (`org_scope.*`), text index on name/contact for support search.
- TTL indexes only for transient collections (sessions, notification queues) — never on ledger/audit.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

5. Event Model & Ingestion

5.1 Universal event envelope

`event_id`, `event_type`, `occurred_at` / `received_at`, `source`, `member_id` / `association_id`, org scope, `amount` / `quantity` / `currency`, `sku` / `category`, `location` / `device`, `idempotency_key`, `correlation_id`, `payload`, `version`.

5.2 Ingestion channels

REST API (real-time), file/SFTP batch, QR/barcode scan simulator, manual entry, referral/activity events.

5.3 Pipeline

Receive → schema validate (Pydantic) → idempotency check (DB unique + Redis fast path) → stage → async rule evaluation → ledger posting → downstream (tier, campaign budget, notification) → analytics stream.

5.4 Idempotency & replay protection

Every event carries an `idempotency_key`. Duplicate keys are rejected at DB layer and logged as `duplicate_rejected` (demonstrable in the event monitor). Offline POS events cached and replayed with the same key — no double accrual.

5.5 Ordering & consistency

Per-account ledger posting is serialized (per-account lock via Redis or Mongo transaction) to prevent race conditions on balance (§7.3).

6. Rules Engine

6.1 Model

Declarative condition/action rules stored as data (not code). No-code/low-code builder in Admin.

6.2 Rule dimensions

Organization, Member, Geography, Product/SKU, Transaction (value/volume/threshold), Time (periodic/milestone), Behavior (activities).

6.3 Lifecycle

draft → validate → **simulate** → approve (maker-checker) → publish → monitor → amend/pause/retire. All versioned and effective-dated.

6.4 Evaluation

Deterministic evaluator returns matched rule(s), computed points, and an **explanation object** (which rule version, which conditions matched, which multipliers applied). Explainability is a hard requirement — the demo must show "why did this transaction earn X."

6.5 Conflict resolution

Configurable stacking, priority ordering, exclusivity, and max-benefit caps. Evaluator resolves conflicts per configured policy and records the decision trace.

6.6 Simulation

Run rules against seeded/historical events without posting to ledger; show projected impact and budget consumption before publish.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

7. Ledger Engine (Financial-Grade Integrity)

7.1 Append-only model

Ledger entries are immutable. Corrections happen via new `reversal` / `adjustment` entries that reference the original — never edits/deletes.

7.2 Tamper-evidence (hash-chaining)

Each entry stores `prev_hash` (hash of prior entry for that account) and `entry_hash = H(entry_body + prev_hash)`. Any retroactive tampering breaks the chain and is detectable by a verification job. This gives Postgres-grade auditability on MongoDB.

7.3 Consistency & concurrency (the MongoDB call, explained)

- Balance mutations use **MongoDB multi-document ACID transactions** (requires replica set) so event consumption + ledger append + balance update commit atomically.
- Per-account **serialization** (advisory lock in Redis keyed by `account_id`, or transaction with retry on write conflict) prevents the classic double-redeem race: two concurrent redemptions cannot both pass a balance check.
- **Trade-off vs Postgres:** Mongo transactions carry a slight performance cost and require careful retry handling on transient write conflicts. For single-tenant loyalty volumes this is well within limits. If transaction rates ever exceed replica-set transaction throughput, we shard by `account_id` (documented scale path).

7.4 Point lifecycle

accrual → pending → posted → (burn | transfer | expiry | reversal | adjustment). Expiry runs as a scheduled job (configurable FIFO/date rules), posts debit entries, updates liability.

7.5 Liability tracking

Running loyalty liability per program, reconciled against catalogue/settlement (§16). Finance dashboard reads from here.

7.6 Manual adjustments

Always require maker-checker (§15) and write a fully-attributed ledger entry with reason, actor, before/after balance.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

8. Tier & Benefits Engine

- Configurable tier definitions and qualification criteria (sales, points, frequency, growth, product mix, tenure, combined).
- Scheduled + event-triggered evaluation. Grace periods and downgrade protection configurable.
- `tier_history` maintained for audit. Tier-specific multipliers, privileges, and dynamic pricing.
- **Demo:** qualifying event → automatic upgrade → member-facing change reflected in portal.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

9. Rewards, Catalogue & Redemption

- Catalogue types: vouchers, merchandise, cashback, donation, partner-burn, points+cash co-pay.
- Redemption flow: eligibility check → balance hold (pending debit) → OTP confirm → vendor fulfilment (mocked adapter with real interface) → success posts burn / **failure auto-refunds via ledger reversal**.
- Inventory management, order tracking, refund workflows.
- **Demo:** successful voucher redemption + a seeded vendor-failure order that auto-refunds points.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

10. Campaign, Promotion & Rules

- No-code campaign builder: setup → audience/conditions → reward/conflict → simulate → approve → publish.
- Budgets with real-time consumption tracking; effective dating; cohort targeting.
- Conflict handling shared with rules engine (§6.5).
- **Demo:** Gold/Platinum dealer double-points campaign on selected SKUs + territory, simulated, approved, published, then a live qualifying transaction updates campaign budget.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

11. Gamification Engine

- Missions, badges, streaks, milestones, leaderboards, quiz/survey mechanics.
- Reward triggering integrates with ledger + rules. Progress tracked in `member_progress`.

12. Engagement & Journey Orchestration

- Segmentation + journey builder (30/60/90-day flows, dormant reactivation).
- Channels: SMS, WhatsApp, push, email, in-app banners (dispatch via integration adapters).
- **Demo:** dormant-user journey triggers a targeted message.
- AI nudges in v1: rule/segment-driven, not ML (illustrative). Flagged as future enhancement.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

13. Analytics & Intelligence

- Operational dashboards, RFM/CLTV, cohort analysis, campaign ROI, loyalty liability, Member 360.
- v1 stance: **real drill-down to evidence** (click a metric → underlying ledger/event records) is mandatory; ML scoring is illustrative on seed data.
- Exportable reports (CSV/PDF) with audit of who exported what (PII governance, §18).

14. Integration & API Platform

14.1 API design

REST/JSON, OpenAPI-documented, versioned (`/v1`), OAuth2-scoped, idempotent writes, cursor pagination, standard error envelope, per-client rate limiting.

14.2 API domains

/auth /users /members /kyc /associations /masters /programs /events /transactions /campaigns /rules /simulations /points/accounts /ledger /tiers /catalog /redemptions /orders /gamification /journeys /notifications /reports /integrations /approvals /tickets /audit /reconciliation

14.3 External integration categories (mocked adapters, real interfaces in v1)

POS/core systems, digital channels, CRM/Salesforce, bank/NBFC/payment partner, government/KYC verification, voucher/rewards aggregators, warehouse/logistics, WhatsApp/SMS/push/email, SSO/IdP, data lake/BI.

14.4 Integration controls

Credential vaulting, correlation/idempotency IDs, retry with backoff, dead-letter queue, control totals for batch, integration health dashboard, full request/response logging (with PII masking).

15. Workflow & Approvals (Maker-Checker)

- Configurable approval matrices keyed by request type, threshold, risk, org node.
- Sequential/parallel approvals, delegation, escalation, SLA timers.
- Applied to: manual point adjustments, high-value redemptions, campaign publish, KYC approvals, rule publish.
- **Demo:** manual point correction → checker approval → new ledger entry + full audit trail.

16. Fraud, Reconciliation & Financial Controls

- **Fraud/velocity:** duplicate invoice detection, repeated QR abuse, device/account velocity thresholds, location anomaly, suspicious redemption patterns → flag/hold/route to review.
- **Reconciliation:** scheduled runs comparing ledger liability vs settlement/catalogue/vendor records; discrepancies raise exceptions.
- **Dispute management:** log/track/resolve member disputes (missing/incorrect points, redemption failures) with audit.

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai

17. Security Architecture

17.1 Authentication

- Admin: SSO via OIDC/SAML against client IdP + mandatory **MFA (TOTP)**.
- Members: phone/email + OTP or password; optional social; refresh-token rotation.
- JWT access tokens (short TTL) + refresh tokens (rotating, revocable via Redis blacklist).

17.2 Authorization (RBAC + data scope)

- Roles: Super Admin, Program Admin, Program Manager, Finance/Loyalty Controller, Compliance/Risk Approver, Operations/Fulfilment, Support Agent, Sales/Field, Partner/Vendor, Member, Auditor (read-only).
- **Attribute/scope-based enforcement:** every request scoped to permitted org hierarchy nodes; partner/vendor users see only assigned data. Enforced at service layer, not just UI.

17.3 Data protection

- **In transit:** TLS 1.2+ everywhere, HSTS.
- **At rest:** MongoDB encryption-at-rest, encrypted object storage, encrypted backups. Field-level encryption for high-sensitivity PII (KYC identifiers, bank details) using a KMS-managed key.
- Secrets in a vault (HashiCorp Vault / cloud KMS), never in code/config.

17.4 Application security

- Input validation (Pydantic), output encoding, parameterized queries (no injection surface via ORM/driver), CSRF protection on session flows, CORS allow-list.
- API rate limiting + throttling (Redis), WAF at edge.
- Dependency scanning (SCA), SAST in CI, secret scanning, container image scanning.
- OWASP Top 10 + API Security Top 10 addressed as a checklist gate before release.

17.5 Audit & immutability

- Every business-critical state change writes an immutable `audit_log` entry: actor, timestamp, action, before/after state, correlation ID.
- Ledger hash-chaining (§7.2) + periodic chain-verification job.
- Auditor role has read-only access to logs and evidence.

17.6 Observability & incident response

- Centralized logging with PII masking, alerting on anomalies (failed logins, velocity breaches, chain-verification failures).

- Documented incident response + breach notification runbook (ties to §18 DPDP obligations).

18. Compliance (India — DPDP Act 2023 + Sectoral)

18.1 Governing framework

India's **Digital Personal Data Protection Act, 2023 (DPDP)** is the primary driver (India-only deployment). GDPR is out of scope but the architecture is GDPR-compatible if geography expands later.

18.2 Data classification

- Classify all fields: Public / Internal / PII / Sensitive PII (KYC IDs, financial, biometric if any).
- PII/Sensitive fields tagged in schema, drive encryption, masking, access control, and retention rules.

18.3 DPDP obligations implemented

- **Consent management:** purpose-specific, granular consent capture with timestamped records; consent withdrawal supported. Consent artifacts stored per member (`members.consent[]`).
- **Notice:** clear notice at collection points in member portal.
- **Data principal rights:** access, correction, erasure, grievance redressal — exposed via member portal + support workflow. Erasure honored via anonymization (ledger/financial records retained in anonymized form to preserve audit/financial integrity — documented lawful basis).
- **Data minimization & purpose limitation:** collect only what's needed per program; enforced via config.
- **Retention:** configurable retention/archival/anonymization policies per data class; scheduled purge jobs with audit.
- **Grievance officer / consent manager:** contact and workflow defined.
- **Breach notification:** runbook to notify Data Protection Board + affected principals within required timelines.

18.4 Data residency

All primary data, replicas, backups, and object storage hosted in an **India region**. No cross-border transfer of personal data in v1. Third-party processors (SMS/WhatsApp/KYC vendors) must be India-region or contractually compliant.

18.5 KYC/AML & financial boundaries (flagged for client decision)

- If the program includes **cashback to bank / direct money movement**, it triggers RBI/payment-system considerations and likely a licensed payment partner + KYC/AML obligations. This spec treats money movement as an **integration boundary** (mocked in v1). **Decision required from client before any real fund flow is built.**
- KYC (e-KYC / video KYC / document verification) integrated via provider adapter; KYC artifacts encrypted, access-controlled, retention-bound.

18.6 Audit & evidence for compliance

Immutable audit log + ledger chain provide demonstrable evidence for regulatory or internal audit.

19. Non-Functional Requirements

- **Availability:** target 99.9% (contractual); graceful degradation (e.g., analytics can lag while ledger stays consistent).
- **Performance:** interactive screens < 2–3s; event ingestion async with defined throughput SLA; API p95 targets per endpoint.
- **Consistency:** strong consistency for ledger/redemption (ACID); eventual consistency acceptable for analytics.
- **Scalability:** horizontal scaling of stateless services; async workers scale independently; documented shard-by-account path for ledger if needed.
- **Localization:** English + Hindi/regional language support in member comms and catalogue.
- **Accessibility:** WCAG 2.1 AA for web interfaces.
- **Maintainability:** business behavior configurable without code releases.
- **Testability:** simulatable rules; automatable API/integration/regression tests.
- **DR:** defined RPO/RTO, automated backups (India region), tested restore, replica-set failover.

20. DevOps & Deployment (Single-Tenant)

- **Environments:** dev → staging → prod, isolated for the single client.
- **CI/CD:** automated pipeline with SAST/SCA/secret scanning gates, container build/scan, automated tests, blue-green or rolling deploy.
- **Infra as code** (Terraform) for reproducibility.
- **MongoDB:** 3-node replica set (transaction support + HA), automated encrypted backups, PITR.
- **Monitoring/alerting:** Prometheus/Grafana, log aggregation, uptime + security alerts.
- Right-sized for one tenant — avoid premature microservice/multi-region complexity.

21. Testing Strategy

- Unit, contract (API), integration (adapters), **rule simulation tests**, **ledger invariant tests** (balance = sum of entries; chain integrity), concurrency/race tests (double-redeem), security tests (authz scope, injection), performance/load, regression, UAT against demo scenarios.
- Ledger invariants and idempotency are the highest-priority test suites — they protect financial integrity.

— End of Technical Baseline v1.0 —

Anmol.berry@fundle.ai · Deepak.tyagi@fundle.ai