Internal Admin Module

LoyalNest App

Overview

This document details the internal admin module for the LoyalNest App, enabling platform-level management of merchants, points, integrations, analytics, customer data imports, GDPR/CCPA compliance, and rate limit monitoring, with RBAC for Shopify Plus multi-user access (50,000+ customers, 1,000 orders/hour). It integrates with admin, analytics, auth, and points services, using PostgreSQL, Redis Streams, and Nx monorepo management for scalability and maintainability. Enhancements include scoped RBAC, real-time alerting, merchant timelines, undo actions, additional KPIs, queue monitoring, predictive suggestions, and QA APIs to improve reliability, usability, and Shopify Plus readiness.

Features

- **Overview** (Admin Service):
  - Display metrics: Merchant count, points issued/redeemed, referral ROI, RFM segments (e.g., At-Risk, Champions), customer import status, rate limit usage, median points adjustment latency, RFM config-to-export rate, import retry success rate.
  - Chart.js visualizations for trends (e.g., points issued, RFM segment counts, import success rate, points adjustment latency), with drag-and-drop customization.
  - Predictive analytics: Merchant churn prediction via gRPC `/analytics.v1/AnalyticsService/PredictChurn` using `orders` and `rfm_segment_counts`.
  - Rule-based suggestions: e.g., “Plan limit will be exceeded in 7 days” based on PostHog usage data, displayed in `AdminOverview.tsx`.
  - APIs: `GET /admin/v1/overview`, gRPC: `/admin.v1/AdminService/GetOverview`.
  - Success Metric: 90%+ query performance under 1s, 20%+ dashboard customization rate, 80%+ suggestion engagement rate.
- **Merchants** (Admin Service):
  - List: `merchant_id`, `shopify_domain`, `plan_id`, `status` (active, suspended, trial), `staff_roles` (JSONB), `language` (JSONB).
  - Search: Fuzzy matching by domain, name.
  - Actions: View details, activate/suspend, adjust points, GDPR export/delete, bulk plan upgrades/downgrades, customer import, undo bulk plan changes/points adjustments (stored in `audit_logs` with `reverted` flag).

- Timeline: Chronological view of merchant actions (e.g., imports, config changes, points adjustments) in `MerchantsPage.tsx` using Chart.js/Polaris.
- Per-merchant rate limit tracking: Shopify API and integration usage, cached in Redis Streams (`admin:rate_limits:{merchant_id}` , TTL 1h), visualized with Chart.js.
- APIs: `GET /admin/v1/merchants` , `POST /admin/v1/merchants/search` , `POST /admin/v1/merchants/{id}/adjust-points` , `POST /admin/v1/merchants/bulk` , `POST /admin/v1/merchants/{id}/undo` , gRPC: `/admin.v1/AdminService/ImportCustomers` .
- Error Handling: `INVALID_MERCHANT_ID` , `UNAUTHORIZED` , `RATE_LIMIT_EXCEEDED` (429), `UNDO_NOT_PERMITTED` .
- RBAC: `admin:full` for all actions, `admin:support` for view/search, `admin:points` for points adjustments, scoped roles (e.g., `admin:merchants:view:shopify_plus` , `admin:merchants:edit:plan`).
- Success Metric: 95%+ action success rate, 80%+ timeline usage rate.

- **Admin Users** (Admin Service, Auth Service):
  - Add/edit/delete users in `admin_users` (username, email ENCRYPTED, password, metadata JSONB with RBAC roles: `admin:full` , `admin:analytics` , `admin:support` , `admin:points` , `admin:merchants:view:shopify_plus` , `admin:merchants:edit:plan`).
  - MFA via Auth0, session timeout (30 min), IP whitelisting (internal IPs).
  - Anomaly detection: Flag unusual actions (e.g., >100 points adjustments/hour, >3 rate limit breaches/hour), log to Sentry (`admin_action_anomaly`), notify `admin:full` via PagerDuty/Opsgenie (AWS SNS webhook).
  - RBAC for Plus: Multi-user access with role-based restrictions (e.g., `admin:analytics` for read-only analytics, `admin:full` for imports, campaigns, user management).
  - APIs: `POST /admin/v1/users` , `PUT /admin/v1/users/{id}` , `DELETE /admin/v1/users/{id}` , gRPC: `/auth.v1/AuthService/CreateAdminUser` .
  - Error Handling: `DUPLICATE_USERNAME` , `INVALID_EMAIL` .
  - Success Metric: 100% secure user management.
- **Logs** (Admin Service):
  - View `api_logs` (route, method, status_code, created_at), `audit_logs` (admin_user_id, action, target_table, target_id, created_at, reverted BOOLEAN).
  - Real-time streaming via WebSocket (`/admin/v1/logs/stream`), cached in Redis Streams (`admin:logs:stream:{merchant_id}` , TTL 30m).
  - Actions Logged: `gdpr_processed` , `rfm_export` , `customer_import` , `customer_import_completed` , `campaign_discount_issued` , `tier_assigned` , `config_updated` , `rate_limit_viewed` , `undo_action` .
  - Notification Status Monitoring: Track referral notification status (`sent` , `failed`) in `email_events` , linked to `email_templates` .
  - Filters: Date, route, user, action, notification status (real-time support).
  - Log Replay: Reprocess logs for QA via `POST /admin/v1/logs/replay` (RBAC: `admin:full`).

  - APIs: `GET /admin/v1/logs/api`, `GET /admin/v1/logs/audit`, `GET /admin/v1/logs/stream`, `POST /admin/v1/logs/replay`.
  - Success Metric: 90%+ query performance under 1s.
- **Integration Health** (Admin Service):
  - Monitor Shopify, Klaviyo, Postscript, Square, Lightspeed (`integrations.status`, `last_checked`, `error_details` JSONB).
  - Real-time alerts in dashboard (e.g., “Shopify API down”) and proactive Slack/Email/PagerDuty alerts via AWS SNS, tracked via PostHog (`admin_integration_alert_sent`).
  - APIs: `GET /admin/v1/integrations/health`.
  - Success Metric: 95%+ integration uptime.
- **Rate Limit Monitoring** (Admin Service):
  - Display Shopify API and integration rate limits (2 req/s REST, 40 req/s Plus, 1–4 req/s Storefront) per merchant and endpoint, cached in Redis Streams (`admin:endpoint_limits:{merchant_id}:{endpoint}`, TTL 1h).
  - Visualize usage trends with Chart.js in `RateLimitsPage.tsx`.
  - Alerts for breaches (>3/hour) via PagerDuty/Opsgenie (AWS SNS webhook).
  - APIs: gRPC: `/admin.v1/AdminService/GetRateLimits`.
  - Success Metric: 90%+ rate limit query performance under 1s.
- **Platform Settings** (Admin Service):
  - Configure plans, global settings, RFM thresholds, notification templates (including RTL for `ar`, `he`).
  - APIs: `POST /admin/v1/settings`, `GET /admin/v1/settings`.
  - RBAC: `admin:full` for updates, `admin:analytics` for read-only.
  - Success Metric: 95%+ settings update success.
- **Login as Merchant** (Admin Service):
  - Generate temporary JWT (1-hour expiry) for merchant access, restricted by RBAC.
  - APIs: `POST /admin/v1/merchants/{id}/login`.
  - RBAC: `admin:full` only.
  - Success Metric: 90%+ login success rate.
- **RFM Configuration and Export** (Admin Service):
  - Configure RFM thresholds (Recency 1–5: 7–90 days, Frequency 1–5: 1–10 orders, Monetary 1–5: $50–$2,500+).

- Use `rfm_segment_counts` materialized view for analytics (merchant_id, segment_name, customer_count, last_refreshed).
- Export segments as CSV/JSON, async processing via Bull queues.
- APIs: `POST /admin/v1/rfm/config`, `POST /admin/v1/rfm/export`.
- RBAC: `admin:full`, `admin:analytics` for configuration/export.
- Success Metric: 80%+ config completion, 90%+ export completion under 5s.

- **Customer Data Import** (Admin Service):
  - Async CSV import with validation (email, shopify_customer_id), GDPR compliance (AES-256 encryption for PII).
  - Real-time progress tracking via WebSocket (`/admin/v1/imports/stream`) in `MerchantsPage.tsx` using Polaris `ProgressBar`, logged in `audit_logs` (`customer_import_completed`).
  - APIs: gRPC: `/admin.v1/AdminService/ImportCustomers`.
  - RBAC: `admin:full` only.
  - Log actions in `audit_logs` (`customer_import`, `customer_import_completed`).
  - Success Metric: 95%+ import success rate.
- **Queue Monitoring** (Admin Service):
  - Monitor Bull queues (retry count, DLQ status, time-in-queue) for RFM exports, customer imports.
  - UI in `QueuesPage.tsx` with Polaris components, showing queue metrics (e.g., jobs in queue, retry count, DLQ status).
  - APIs: `GET /admin/v1/queues`.
  - Success Metric: 95%+ queue operation success rate.
- **Event Simulation** (Admin Service):
  - Simulate campaign/referral events for QA via `POST /admin/v1/events/simulate` (e.g., trigger `campaign_discount_issued`, `referral_sent`).
  - APIs: `POST /admin/v1/events/simulate`.
  - RBAC: `admin:full` only.
  - Success Metric: 95%+ simulation success rate.

Technical Details

- **Backend** (Admin Service, Auth Service):

- NestJS for APIs, gRPC for inter-service communication (Admin ↔ Analytics, Admin ↔ Points, Admin ↔ Auth).
- Shopify GraphQL Admin API for merchant/customer data.
- Bull queues for async tasks (exports, customer imports, RFM calculations), 5 retries (2s initial delay), with DLQ and `/admin/v1/queue/reprocess` endpoint (RBAC: `admin:full`).
- Circuit breakers (`nestjs-circuit-breaker`) for gRPC calls (e.g., `/admin.v1/AdminService/GetOverview`, `/auth.v1/AuthService/CreateAdminUser`, `/admin.v1/AdminService/ImportCustomers`).
- Error Handling: `UNAUTHORIZED`, `INVALID_INPUT`, `QUEUE_FULL`, `RATE_LIMIT_EXCEEDED`, `UNDO_NOT_PERMITTED`.
- gRPC Endpoints:
  - `/admin.v1/AdminService/GetOverview`
  - `/admin.v1/AdminService/ImportCustomers`
  - `/admin.v1/AdminService/GetRateLimits`
  - `/auth.v1/AuthService/CreateAdminUser`

- **Frontend** (Frontend Service):
  - React components (`AdminOverview.tsx`, `MerchantsPage.tsx`, `LogsPage.tsx`, `UsersPage.tsx`, `IntegrationsPage.tsx`, `RateLimitsPage.tsx`, `QueuesPage.tsx`) using Vite, Polaris, Tailwind CSS.
  - Rate limit monitoring screen with real-time Shopify API and per-endpoint limits, visualized with Chart.js.
  - Merchant timeline in `MerchantsPage.tsx` using Chart.js/Polaris for action history.
  - Queue monitoring in `QueuesPage.tsx` with Polaris components for Bull queue metrics.
  - Dynamic locale detection (`navigator.language`) with Polaris `Select` override, cached in Redis (`admin:locale:{user_id}`, TTL 7d).
  - i18next for multilingual support (`en`, `es`, `fr`, `ar` with RTL via `dir=auto`), WCAG 2.1 compliant.
- **Database**:
  - `admin_users` (id, username UNIQUE, email ENCRYPTED, password, metadata JSONB with RBAC roles: `["admin:full", "admin:analytics", "admin:support", "admin:points", "admin:merchants:view:shopify_plus", "admin:merchants:edit:plan"]`)
  - `api_logs` (id, merchant_id, route, method, status_code, created_at)
  - `audit_logs` (id UUID, admin_user_id, action CHECK('gdpr_processed', 'rfm_export', 'customer_import', 'customer_import_completed', 'campaign_discount_issued', 'tier_assigned', 'config_updated', 'rate_limit_viewed', 'undo_action'), target_table, target_id, created_at, reverted BOOLEAN)
  - `integrations` (integration_id, merchant_id, type CHECK('shopify', 'klaviyo', 'postscript', 'square', 'lightspeed'), status CHECK('ok', 'error'), settings, api_key ENCRYPTED)

  - `rfm_segment_counts` (merchant_id, segment_name, customer_count, last_refreshed)
  - `email_events` (event_id, merchant_id, event_type CHECK('sent', 'failed'), recipient_email ENCRYPTED)
  - Indexes: `admin_users(username, email)`, `api_logs(merchant_id, route)`, `audit_logs(admin_user_id)`, `integrations(merchant_id)`, `rfm_segment_counts(merchant_id)`
  - Partitioning: `api_logs`, `audit_logs` by `merchant_id`.
- **Caching**: Redis Streams (`admin:metrics:{period}`, TTL 6h; `admin:logs:{merchant_id}`, `admin:logs:stream:{merchant_id}`, TTL 30m; `admin:rate_limits:{merchant_id}`, `admin:endpoint_limits:{merchant_id}:{endpoint}`, TTL 1h; `admin:locale:{user_id}`, TTL 7d) for overview, logs, integration health, rate limits, locale.
- **Feature Flags**: LaunchDarkly for “Login as Merchant”, RFM export, integration health, customer import, queue monitoring, event simulation.

Integrations

- **Shopify**: GraphQL Admin API, `customers/data_request`, `customers/redact` webhooks (5 retries).
- **Klaviyo/Postscript/Square/Lightspeed**: Monitor health via API calls (`/v2/payments`, `/api/2.0/sales`, `/sms/messages`), track notification status in `email_events`, proactive alerts via AWS SNS (Slack, Email, PagerDuty/Opsgenie).
- **Rate Limiting**: 2 req/s (REST), 40 req/s (Plus), 1–4 req/s (Storefront).

Performance Optimizations

- **Redis Streams**: Cache metrics, logs, integration health, rate limits (`admin:rate_limits:{merchant_id}`, `admin:endpoint_limits:{merchant_id}:{endpoint}`).
- **PostgreSQL**: Connection pooling for 10,000+ queries, partitioning for logs, materialized views for `rfm_segment_counts`.
- **Load Testing**: k6 for 5,000 merchants, 50,000 customers, Black Friday surges (10,000 orders/hour, 100 concurrent admin actions).

Security and Compliance

- **GDPR/CCPA**: Encrypt `admin_users.email`, `integrations.api_key` with pgcrypto. Cascade deletes for `customers/redact`. Log `gdpr_processed` in `audit_logs`. 90-day backup retention for `audit_logs`, `api_logs`.
- **Recovery Plan**: PostgreSQL RDS point-in-time recovery (<1h downtime), Redis AOF persistence with daily S3 backups. Test with Chaos Mesh in Kubernetes.

- **Audit Logging**: Log all admin actions (points adjustments, user management, RFM config/export, customer import, campaign discounts, rate limit views, undo actions).
- **RBAC**: Enforce for all actions:
  - `admin:full` : All actions (user management, points adjustments, imports, campaigns, RFM config/export, rate limit monitoring, queue monitoring, event simulation).
  - `admin:analytics` : Read-only for analytics, RFM config/export.
  - `admin:support` : Read-only for merchants, logs, integration health.
  - `admin:points` : Points adjustments, campaign management, VIP tiers.
  - `admin:merchants:view:shopify_plus` : View Shopify Plus merchant details.
  - `admin:merchants:edit:plan` : Bulk plan upgrades/downgrades, undo actions.
- **Security**: JWT (1-hour expiry), MFA (Auth0), IP whitelisting, session timeout (30 min), anomaly detection for admin actions, rate-limiting for `/admin/v1/*` (100 req/min per user).
- **Backup**: 90-day retention for `audit_logs` , `api_logs` , with recovery plan.

Testing

- **Unit**: Jest for APIs, RBAC logic (including scoped roles), export processing, customer import, rate limit monitoring, anomaly detection, queue monitoring, event simulation, undo actions.
- **Integration**: Shopify, Klaviyo, Postscript, Square, Lightspeed APIs, customer import workflows, queue monitoring, event simulation.
- **E2E**: Admin dashboard, login as merchant, logs, exports, rate limit monitoring, customer import, queue monitoring, event simulation, merchant timeline, undo actions (Cypress).
- **Load Test**: k6 for 5,000 merchants, 50,000 customers, Black Friday surges (10,000 orders/hour, 100 concurrent admin actions).
- **Penetration Test**: OWASP ZAP for `/admin/v1/*` and gRPC endpoints, validating RBAC and encryption.
- **Chaos Test**: Chaos Mesh in Kubernetes to simulate Admin Service, Redis, and PostgreSQL failures, validating circuit breakers, DLQs, and fallback UI.

Deployment

- **VPS**: Docker Compose for admin service, auth service, PostgreSQL, Redis, Nginx.

- **CI/CD**: GitHub Actions with Nx change detection, blue-green deployment via AWS ECS Fargate.
- **Monitoring**: Grafana, Prometheus, Sentry for API latency, errors, integration health, rate limits, anomaly detection, queue metrics.

Documentation

- Multilingual admin docs (English, Spanish, French, Arabic with RTL) with 1–2 minute videos for merchant management, logs, integration health, rate limit monitoring, customer import, queue monitoring, event simulation.
- OpenAPI/Swagger for `/admin/v1/*`, gRPC proto files for `/admin.v1/*`, `/auth.v1/*`, developer guide for RBAC, GDPR compliance, recovery plan, queue monitoring, event simulation.

Feedback Collection

- Conduct Typeform survey with 5–10 admin users (2–3 Plus) on dashboard usability, merchant management, rate limit monitoring, queue monitoring, and timeline/undo features.
- Log feedback in Notion and iterate in post-deployment phase.
- Deliverable: Feedback report with Plus-scale insights.

Future Enhancements (Phase 6)

- **Kafka/NATS JetStream**: Replace Redis Streams for log monitoring and rate limit tracking, enabling longer retention and replay support.
- **Versioned Configuration Management**: Track global settings and RFM config as versioned deltas with rollback support (`rfm_config_history` table, `POST /admin/v1/rfm/rollback`).
- **Multi-tenancy**: Support sub-RBAC groups (e.g., per brand for Shopify Plus) and multi-tenant isolation for agencies.
- **AI-Powered Suggestions**: Extend predictive suggestions (e.g., “Segment X has low ROI. Archive it?”) using xAI API, integrated with `/analytics.v1/AnalyticsService`.