Herethere Loyalty App Development Roadmap

Contents

1 Overview

This roadmap outlines the development plan for the `Herethere Loyalty` Shopify app, targeting small and medium-sized merchants (100–10,000 customers, AOV \$20–\$200). The plan is divided into four phases over 18 months, leveraging the tech stack (`NestJS` with TypeScript, `Vite + React`, Rust with Shopify Functions, `PostgreSQL`, `Redis`). It includes features, tasks, enhancements, timelines, success metrics, and ongoing best practices to deliver a production-grade app with **Must Have**, **Should Have**, and **Could Have** features.

2 Phase 1: TVP Development + Internal Admin Module (7 Months)

Goal: Build a production-grade TVP with **Must Have** features: points (purchases, signups, reviews, birthdays), SMS/email referrals (`Twilio`), basic RFM analytics (churn risk), Shopify POS integration (points earning), automated loyalty email flows, data import, and a simplified admin module.

2.1 Enhancements & Best Practices

- Generate `OpenAPI/Swagger` docs for `NestJS` APIs.
- Integrate centralized logging (`Loki/Grafana`) and monitoring (`Prometheus`, `Grafana`) on VPS.
- Add health checks and alerting for `NestJS`, Rust, `Redis`, `PostgreSQL`, `Nginx`.
- Schedule weekly `PostgreSQL`/`Redis` backups and quarterly disaster recovery drills.
- Conduct manual exploratory and accessibility (a11y) testing for dashboard, widget, and admin UI.
- Integrate `PostHog` for feature usage tracking (e.g., RFM wizard completion, referral popup clicks).
- Implement guided onboarding (in-app tooltips, checklists) for merchants.
- Conduct early usability testing of RFM setup wizard and SMS referral popup with 5–10 merchants.
- Review security practices (`OAuth`, `JWT`, GDPR, field-level encryption for `customers.email`, `rfm_score`) bi-monthly.
- Audit `npm`, `cargo`, and Docker dependencies monthly.
- Document infrastructure as code (`Docker Compose`, `Nginx`) in GitHub.
- Allocate 10% buffer time for freelancer coordination and AI code review.
- Maintain developer and merchant documentation with screenshots and 1–2 minute videos for key flows (e.g., RFM setup, points configuration).

2.2 TVP Features

1. *Welcome Page*
   - Setup tasks: Launch program, add widget to theme, configure points, basic RFM.

- Congratulatory messages (e.g., first redemption, first referral).

2. *Program - Points*
   - *Earn*: Purchases (1 point/$), signups (200 points), reviews (100 points), birthdays (200 points).
   - *Redeem*: Discounts (500 points for $5), free shipping (1,000 points), free products (1,500 points), coupons at checkout.
   - *Branding*: Customizable rewards panel, launcher button, points currency (e.g., ”Stars”).
   - *Status*: Enable/disable program.
3. *Program - Referrals*
   - SMS/email referral popup via `Twilio`, rewarding referrer/friend (10% off).
   - Dedicated referral page with incentives for both.
   - Track codes in `referrals` table (pending, completed, expired).
4. *Customers*
   - List with name, email, points balance, RFM segment (e.g., ”At-Risk”).
   - Search by name/email.
5. *Analytics*
   - Metrics: Program members, points transactions, referral ROI, RFM-based churn risk (static thresholds: Recency <90 days, Frequency 1–2, Monetary <$50 for AOV $50).
   - Chart: Bar chart for RFM segments in `AnalyticsPage.tsx` (`Chart.js`).
6. *On-Site Content*
   - SEO-friendly loyalty page, rewards panel, launcher button, points display on product pages.
   - Nudges: Post-purchase prompts, email capture popups.
   - Launchers: Embedded in Shopify checkout and customer accounts.
7. *Settings*
   - Store details, billing (Free: 300 orders, $29/month: 500 orders, $99/month: 1,500 orders).
   - Branding: Basic rewards panel customization.
   - RFM: Static thresholds (e.g., Recency 5: <30 days) in `program_settings.rfm_thresholds` (JSONB).
8. *Shopify Integration*
   - `OAuth` via `@shopify/shopify-app-express`, `orders/create` webhook for points/RFM updates.
   - POS: Points earning (1 point/$).

9. *Other*
   - Automated loyalty email flows (e.g., points earned, redemption reminders).
   - Data import from `Smile.io`, `LoyaltyLion`.
10. *Customer Widget*
    - `React` component for points balance, redemption (discounts, free shipping), SMS/email referral popup, RFM nudges (e.g., ”Stay Active!”).

2.3 Internal Admin Module Features

1. *Overview*
   - Merchant count, points issued/redeemed, referral ROI, RFM segments (`Chart.js`).
2. *Merchants*
   - List (ID, domain, plan), search, view details, adjust points.
3. *Admin Users*
   - Add/edit/delete admins in `admin_users` (`bcrypt`, `JWT`).
4. *Logs*
   - View `api_logs`, `audit_logs` for debugging, RFM changes.

2.4 Database Schema

- *Tables*: `merchants`, `customers` (with `rfm_score` JSONB), `points_transactions`, `referrals`, `rewards`, `reward_redemptions`, `program_settings` (with `rfm_thresholds` JSONB), `shopify_sessions`, `customer_segments`, `admin_users`, `api_logs`, `audit_logs`.
- *Indexes*: `customers(email, merchant_id, rfm_score)`, `points_transactions(customer_id` `referrals(merchant_id)`.
- Use `PostgreSQL` with JSONB for RFM configs; defer `MongoDB` to Phase 4.

2.5 Tasks

1. *Backend (NestJS/TypeScript)*
   - APIs: `/api/points` (purchases, signups, reviews, birthdays), `/api/referral` (SMS/email), `/api/analytics` (RFM segments, churn risk), `/api/data-import` (`Smile.io`/`LoyaltyLion`).
   - Shopify: `OAuth`, `orders/create` webhook for points (1 point/$) and RFM updates (`rfm_score.r`).
   - `Twilio`: SMS/email referrals, storing codes in `referrals` table.
   - Admin APIs: `/admin/merchants`, `/admin/points/adjust`, `/admin/referrals`, `/admin/rfm-segments`, `/admin/logs` with `JWT`.
   - Cache points, referrals, RFM scores in `Redis`.
   - Use AI (`GitHub Copilot`, `Cursor`) for API boilerplate, `Jest` tests; manually review for Shopify compliance.

2. *Backend (Rust/Wasm)*
   - Shopify Functions: Discounts (amount/percentage off), basic RFM score updates.
   - Use Shopify CLI for testing; generate Rust code and `cargo test` cases with AI.
3. *Frontend (Vite + React)*
   - Components: `WelcomePage.tsx` (setup, messages), `PointsPage.tsx` (configure earning/redemption), `ReferralsPage.tsx` (SMS/email config), `CustomersPage.tsx` (list, search), `AnalyticsPage.tsx` (RFM chart), `SettingsPage.tsx` (store, billing, branding), `CustomerWidget.tsx` (points, referrals, nudges).
   - On-Site Content: SEO-friendly loyalty page, rewards panel, launcher button, points display, popups.
   - Admin frontend: Dashboard, merchants, RFM segments, logs.
   - Use AI for components, `Cypress` tests; outsource `Polaris` compliance review ($1,000).
4. *Database*
   - Apply `herethere_full_schema.sql` with JSONB for `rfm_score`, `rfm_thresholds`.
   - Add indexes for performance (e.g., `customers(email, merchant_id)`).
   - Use AI for index optimization.
5. *Deployment*
   - VPS (Ubuntu with Docker) using `Docker Compose` for `NestJS`, `PostgreSQL`, `Redis`, `Vite + React` frontend.
   - `Nginx` for frontend assets and reverse proxy to `NestJS` APIs.
   - `GitHub Actions` for CI/CD (testing, deployment to VPS).
   - Provide `Docker Compose` scripts for VPS setup.
6. *Testing*
   - Unit: `Jest` for `NestJS` APIs, `cargo test` for Rust Functions, `Jest` for RFM logic.
   - Integration: Shopify/`Twilio`/RFM/data-import flows (`Jest`).
   - E2E: Dashboard, widget, popups, RFM UI (`Cypress`).
   - Load test: 5,000 customers with `Redis`/`PostgreSQL`.
   - Outsource QA to freelancer ($2,500) for `Cypress` and exploratory testing.
7. *Shopify App Store*
   - Optimize listing with demo videos (RFM analytics, SMS referrals, $29/month pricing, data import).
   - Ensure `Polaris`, `App Bridge`, GDPR compliance (encrypt `customers.email`, `rfm_score`).

2.6 Timeline

- Months 1–2: Schema, `OAuth`, Points/Referral APIs, static RFM analytics.
- Months 3–4: `Vite + React` dashboard, widget, admin frontend, RFM chart, on-site content (loyalty page, popups).
- Months 5–7: POS integration, Rust Functions, data import, testing, VPS deployment.

2.7 Deliverables

- TVP with **Must Have** features: Points, SMS/email referrals, basic RFM analytics, Shopify POS, email flows, data import.
- Admin module with RFM segment views, merchant management, logs.
- Shopify/`Twilio`/`Yotpo`/`Klaviyo` integrations.
- Test suite, VPS deployment with `Docker` and `Nginx`.
- Merchant documentation with setup guides and videos.

3 Phase 2: Core Feature Expansion + Admin Enhancements (4 Months)

Goal: Add **Should Have** features: VIP tiers (spending-based), advanced RFM configuration, exit-intent popups, `Klaviyo`/`Mailchimp` integration, multi-store point sharing, behavioral segmentation, and enhance admin module.

3.1 Enhancements & Best Practices

- Conduct bi-weekly feedback sessions with beta testers (surveys, interviews, `PostHog` analytics).
- Maintain public changelog for transparency.
- Implement i18n framework for future multilingual support.
- Add `k6` load testing to CI/CD for RFM and VIP tier updates.
- Monitor metrics: RFM wizard completion (80%+), referral conversion (5%+), RFM segment movement (10%+ to higher tiers).
- Define acceptance criteria for outsourced UI/QA work.

3.2 Features

1. *Program - Referrals*
    - Configure rewards for referrer/friend (e.g., points, discounts).
    - Add social sharing (Facebook, Instagram) via referral links.
    - Status toggle (active/disable).
2. *Program - VIP Tiers*
    - Thresholds: Spending or engagement-based (e.g., Silver: $100+, Gold: $500+).
    - Perks: Early product access, birthday gifts, exclusive discounts.

- Track tier changes in `vip_tiers` (with `rfm_criteria` JSONB), notify via email.

3. *Program - Activity*
   - Display points, referrals, VIP tier changes in logs.
   - Filter by customer/date.
4. *Analytics*
   - Reports: Loyalty-driven revenue, redemption rate, RFM tier engagement, repeat purchase rate.
   - `Chart.js` visualizations for tier engagement.
5. *On-Site Content*
   - Exit-intent popups, discount banners, point calculators, checkout extensions.
   - Launchers: Apple/Google Wallet integration.
6. *Settings*
   - Advanced RFM configuration: Thresholds (Recency, Frequency, Monetary), tiers, adjustment frequency (daily/weekly).
   - Setup wizard with pre-filled thresholds (e.g., Recency 5: <30 days, Monetary 5: $250+ for AOV $50).
   - `Chart.js` preview for segment sizes.
7. *Integrations*
   - `Klaviyo` (advanced events), `Mailchimp`, `Yotpo` Email & SMS for personalized campaigns.
   - Email templates for points, rewards, VIP tiers.
8. *Other*
   - Multi-store point sharing.
   - Behavioral segmentation (e.g., purchase frequency, churn risk).
9. *Admin Module Enhancements*
   - Plan upgrades/downgrades, integration health checks, RFM config management.
   - Admin roles (superadmin, support) in `admin_users.metadata`.

3.3 Database Schema

- Add `vip_tiers` (with `rfm_criteria` JSONB), `email_templates`, `email_events`, `integrations`.
- Indexes: `vip_tiers(merchant_id)`, `email_templates(merchant_id)`.
- Sync RFM configs to `customer_segments`, `vip_tiers`.

3.4 Tasks

1. *Backend (NestJS/TypeScript)*
   - APIs: `/api/referrals` (social sharing), `/api/vip-tiers`, `/api/rfm-config`, `/api/activity`, `/api/analytics` (advanced reports).
   - Integrations: `Klaviyo/Mailchimp/Yotpo` APIs for RFM notifications, email campaigns.
   - Cache VIP tiers, RFM configs in `Redis`.
   - Use AI for code, `Jest` tests; manually review.
2. *Backend (Rust/Wasm)*
   - Shopify Functions: VIP multipliers, real-time RFM tier updates.
   - Generate Rust code, `cargo test` cases with AI.
3. *Frontend (Vite + React)*
   - Pages: `ReferralsPage.tsx` (social sharing), `VIPPage.tsx` (tiers), `RFMConfigPage.tsx` (thresholds, wizard), `ActivityPage.tsx`, `IntegrationsPage.tsx`.
   - Update `AnalyticsPage.tsx` with RFM tier reports.
   - On-Site Content: Exit-intent popups, discount banners, point calculators.
   - Admin frontend: Plan management, RFM config view.
   - Use AI for components, `Cypress` tests; outsource `Polaris` review ($1,000).
4. *Database*
   - Add tables for VIP tiers, email templates, integrations.
   - Optimize indexes for RFM and VIP queries.
5. *Testing*
   - Unit: `Jest` for APIs, RFM/VIP logic.
   - Integration: `Klaviyo/Mailchimp/Yotpo`, RFM tier updates.
   - E2E: RFM UI, popups, admin roles (`Cypress`).
   - Load test: 5,000+ customers with `Redis` caching.
6. *Deployment*
   - Optimize `Redis` caching for RFM, VIP, analytics on VPS.
   - Plan AWS ECS migration for future scaling.

3.5 Timeline

- Months 8–9: Social referrals, VIP tiers, `Klaviyo/Mailchimp` integration.
- Months 10–11: Advanced RFM config, activity logs, popups, admin enhancements.

3.6 Deliverables

- **Should Have** features: Social referrals, VIP tiers, advanced RFM, `Klaviyo`/`Mailchimp`, popups, behavioral segmentation, multi-store point sharing.
- Enhanced admin module with RFM config and plan management.
- Updated test suite, VPS deployment with `Docker` and `Nginx`.
- Merchant documentation with RFM setup and integration guides.

4 Phase 3: Advanced Features and Integrations + Admin Polish (4 Months)

Goal: Add **Could Have** features: bonus campaigns, gamification, multilingual support, non-Shopify POS, advanced analytics, developer toolkit, and finalize admin module.

4.1 Enhancements & Best Practices

- Implement load balancer and stateless services for horizontal scaling.
- Overlap testing with Phase 2 feedback to iterate on RFM and referrals.
- Set go/no-go milestone: TVP must achieve 90% merchant satisfaction and 5%+ referral conversion.
- Monitor API latency, error rates, and database performance with `Grafana` dashboards.
- Conduct accessibility (a11y) and localization testing for widget and dashboard.

4.2 Features

1. *Bonus Campaigns*
   - Types: Time-sensitive promotions, goal spend, points multipliers, limited-time bonuses.
   - Conditions: Scheduled via no-code dashboard, tied to purchases.
2. *Gamification*
   - Badges, leaderboards in widget.
3. *On-Site Content*
   - Rewards Sticky Bar, checkout extensions, point calculators.
4. *Integrations*
   - Non-Shopify POS (`Square`, `Lightspeed`).
   - `Gorgias`, `Yotpo` Reviews & UGC, `Postscript`, `Shopify Flow`.
5. *Settings*
   - Multilingual widget (10+ languages with customizable text).
   - Multi-currency discounts.
   - Developer toolkit for Shopify metafields.
6. *Analytics*

- Advanced reports (25+ metrics: ROI, customer behavior, redemption rates).
- Comparisons with similar stores.

7. *Admin Module*
   - Platform settings, integration health, RFM segment export.
   - Advanced analytics dashboard (`Chart.js`).

4.3 Database Schema

- Add `bonus_campaigns`, `gamification_achievements`, `nudges`, `nudge_events`.
- Indexes: `bonus_campaigns(merchant_id)`, `nudges(merchant_id)`.
- Update `merchants.language` for multilingual support.

4.4 Tasks

1. *Backend (NestJS/TypeScript)*
   - APIs: `/api/campaigns`, `/api/gamification`, `/api/nudges`, `/api/integrations`.
   - Integrations: `Gorgias`, `Yotpo`, `Postscript`, `Square`, `Lightspeed`.
   - Use AI for code, `Jest` tests.
2. *Backend (Rust/Wasm)*
   - Shopify Functions: Campaign discounts, gamification rewards.
3. *Frontend (Vite + React)*
   - Pages: `CampaignsPage.tsx`, `GamificationPage.tsx`, `SettingsPage.tsx` (multilingual, developer tools).
   - Update `AnalyticsPage.tsx` with advanced reports.
   - On-Site Content: Sticky Bar, checkout extensions.
   - Admin frontend: Integration health, RFM export.
4. *Testing*
   - Unit: `Jest` for APIs, campaign/gamification logic.
   - Integration: Non-Shopify POS, `Gorgias`/`Yotpo`.
   - E2E: Multilingual widget, advanced analytics (`Cypress`).
   - Load test: 10,000 customers.
5. *Deployment*
   - VPS with `Docker`, `Nginx`, `Cloudflare CDN` for multilingual content.
   - Optimize `PostgreSQL` with partitioning for `points_transactions`, `referrals`.

4.5 Timeline

- Months 12–13: Bonus campaigns, gamification, Sticky Bar.
- Months 14–15: Non-Shopify POS, multilingual support, advanced analytics.

4.6 Deliverables

- **Could Have** features: Bonus campaigns, gamification, multilingual widget, non-Shopify POS, advanced analytics, developer toolkit.
- Polished admin module with integration health and RFM export.
- Updated test suite, VPS deployment with `CDN`.
- Merchant documentation with multilingual and campaign guides.

5 Phase 4: Optimization and Scaling (3 Months)

Goal: Scale for 5,000+ merchants, achieve Built for Shopify certification, iterate based on feedback.

5.1 Enhancements & Best Practices

- Test against Shopify API sandbox to catch breaking changes.
- Monitor Shopify developer changelogs and webhooks.
- Maintain runbook for `Docker`, `Nginx`, `Redis` restarts on VPS.
- Validate `PostgreSQL`/`Redis` backups monthly.
- Iterate on RFM usability, SMS referrals, and gamification based on `PostHog` analytics.

5.2 Tasks

1. *Optimization*
    - `NestJS`: Optimize APIs with async/await for concurrency.
    - Rust: Transition RFM analytics to Rust for performance.
    - `Redis`: Cache points, referrals, RFM, gamification.
    - `PostgreSQL`: Partition `points_transactions`, `referrals`, `vip_tiers`.
2. *Analytics Enhancements*
    - RFM reports (tier engagement, redemption rate, churn reduction) with `Chart.js`.
3. *Shopify Certification*
    - Ensure `Polaris`, GDPR compliance, load test for 5,000+ customers.
4. *User Feedback*
    - Iterate on RFM wizard, SMS referrals, gamification with 20–30 merchants.
5. *Marketing*

- Promote via Shopify Reddit/Discord, ads, case studies (e.g., 15% churn reduction, 10% referral conversion).

6. *VPS Maintenance*
   - Monitor `Docker` containers, `Nginx`, `Redis` performance.
   - Update `Docker Compose` scripts for new features.

5.3 Timeline

- Months 16–18: Optimization, certification, feedback iteration.

5.4 Deliverables

- Scalable infrastructure for 5,000+ merchants on VPS.
- Advanced RFM analytics with detailed reports.
- Built for Shopify certification application.
- Marketing strategy for 100+ merchants.
- Updated VPS maintenance guide.

6 Full Roadmap Timeline

Phase	**Duration**	**Main Tasks**
Phase 1	7 months	Schema, OAuth, Points/Referral APIs, RFM analytics, POS, data import, admin module, VPS deployment
Phase 2	4 months	Social referrals, VIP tiers, advanced RFM, Klaviyo/Mailchimp, popups, behavioral segmentation
Phase 3	4 months	Bonus campaigns, gamification, multilingual widget, non-Shopify POS, advanced analytics
Phase 4	3 months	Optimization, Shopify certification, feedback iteration, marketing
Total	**18 months**	

Table 1: Full Roadmap Timeline

7 Success Metrics

- *Phase 1*: 90% merchant satisfaction in beta, 80% RFM wizard completion, 5%+ SMS referral conversion.
- *Phase 2*: 10%+ RFM tier engagement, 5%+ social referral conversion.
- *Phase 3*: 20%+ repeat purchase increase, 15%+ gamification engagement.

- *Phase 4*: 100+ merchants in 6 months, 4.5+ star rating, Built for Shopify certification in 12 months.

8 Next Steps

- *Month 1 Sprint*:
  - Set up `PostgreSQL` with `herethere_full_schema.sql` (JSONB fields, indexes).
  - Implement Shopify `OAuth` using `@shopify/shopify-app-express` in `NestJS`.
  - Generate `/api/points`, `/api/referral` APIs and `Jest` tests with AI; review manually.
  - Join Shopify Reddit/Discord to recruit 5–10 beta testers.
  - Set up VPS with `Docker`, `Nginx`, `GitHub Actions` for CI/CD.
- *Seek Feedback*: Share TVP prototype with 3–5 merchants by Month 3, focusing on RFM usability and SMS referral effectiveness.
- *Learning Plan*: Complete 2-hour `NestJS` (e.g., NestJS Crash Course) and `Vite + React` (e.g., freeCodeCamp) tutorials in Week 1 to understand AI-generated code.

9 Ongoing Best Practices & Metrics

- Maintain up-to-date developer and merchant documentation with videos.
- Publish public changelog for transparency.
- Monitor API latency, error rates, database performance via `Grafana`.
- Review Shopify security/API best practices monthly.
- Plan for `Kubernetes` migration as merchant count exceeds 5,000.
- Track `PostHog` metrics: RFM wizard completion, referral conversion, tier engagement.
- Optimize `PostgreSQL` (partitioning) and `Redis` (cache invalidation).
- Document infrastructure as code (`Docker Compose`, `Nginx`).
- Set up `Grafana` dashboards for real-time monitoring.
- Conduct monthly security/dependency audits.
- Ensure accessibility (a11y) and localization for all user-facing features.
- Maintain runbook for VPS operations (e.g., `Docker` restarts, `Redis` failover).