Herethere Loyalty App: Tech Stack Summary

Contents

1 Overview

This document summarizes the tech stack for the Herethere Loyalty Shopify app, optimized for a solo beginner developer using AI-driven development to avoid project failure. The stack reflects the preference for NestJS from the start, Vite + React for the frontend, and deployment on a VPS instead of Railway, as outlined in the Project Plan and Roadmap. It supports scalability for 5,000+ customers and Shopify compliance.

2 Backend

2.1 Framework: NestJS (TypeScript)

- *Why*: NestJS's modular structure (controllers, services, modules) organizes APIs (/api/points, /api/referral, /api/analytics, /admin/*) and admin module logic for a medium-sized app. TypeScript ensures type safety for critical features like RFM scores (customers.rfm_sco and points transactions, reducing bugs. It supports scalability for Phase 2 (RFM tiers, Klaviyo integrations) without refactoring. Integrates with @shopify/shopify-app-express for Shopify OAuth and webhooks.
- *Use Case*: APIs for points (signup, birthday), SMS referrals (Twilio), basic RFM analytics (static thresholds), and admin module (/admin/merchants). Handles Shopify integrations (OAuth, orders/create webhook) and Twilio SMS.
- *AI Assistance*: Generate NestJS controllers, services, and Jest tests (e.g., "Write a NestJS controller for SMS referrals") and explain concepts (e.g., decorators).

2.2 Additional Backend: Rust/Wasm

- *Why*: Used for Shopify Functions (discounts, basic RFM score updates, e.g., rfm_score.r on orders/create), ensuring high performance for real-time logic.
- *Use Case*: Implements amount/percentage discounts (e.g., 500 points for $5 off) and updates RFM scores in customers.rfm_score (JSONB).
- *AI Assistance*: Provide Rust code and cargo test cases (e.g., "Write a Rust Shopify Function for discounts") and guide Shopify CLI setup.

2.3 Database: PostgreSQL (JSONB)

- *Why*: Consolidates data storage for TVP, using JSONB for flexible RFM configs (rfm_score, rfm_thresholds) and tables (merchants, customers, points_transactions, referrals, etc.), as per herethere_full_schema.sql. Indexes on customers(email, merchant_id, rfm_score), points_transactions(customer_id), and referrals(merchant_id) ensure performance for 5,000+ customers.
- *Use Case*: Stores customer data, points transactions, referral codes, and RFM segments. JSONB simplifies static RFM thresholds (e.g., Recency 5: <30 days).
- *AI Assistance*: Optimize indexes (e.g., "Suggest PostgreSQL indexes for RFM queries") and generate schema scripts.

2.4 Caching: Redis

- *Why*: Caches points balances, referral codes, and RFM scores/segments to support scalability (5,000+ customers).

- *Use Case*: Reduces database load for frequent queries (e.g., RFM segment counts in `AnalyticsPage.tsx`).
- *AI Assistance*: Provide `Redis` integration code (e.g., "Add `Redis` caching to `NestJS` for points").

3 Frontend

3.1 Framework: Vite + React (TypeScript)

- *Why*: `Vite`'s fast builds and hot module replacement (HMR) accelerate development of `Polaris`-compliant components (`WelcomePage.tsx`, `PointsPage.tsx`, `AnalyticsPage.tsx`, `CustomerWidget.tsx`). `TypeScript` ensures consistency with the backend and type-safe props (e.g., `interface Customer { points: number; }`). Integrates with Shopify's `Polaris`, `App Bridge`, and `Tailwind CSS` for Shopify compliance.
- *Use Case*: Builds merchant dashboard (points, referrals, RFM analytics with `Chart.js`), customer widget (points balance, SMS referral popup), and admin module (merchant management, RFM segments). Supports mobile-friendly design and Phase 3 features (gamification, multilingual).
- *AI Assistance*: Generate JSX components, `Tailwind` styles, and `Cypress` tests (e.g., "Write a `Polaris`-compliant `React` component for RFM chart").

3.2 UI Framework: Shopify Polaris

- *Why*: Ensures Shopify App Store compliance with a consistent, merchant-friendly UI for dashboard and admin module.
- *Use Case*: Used for all frontend pages and customer widget.

3.3 Styling: Tailwind CSS

- *Why*: Provides utility-first styling for rapid UI development, complementing `Polaris`.
- *Use Case*: Styles dashboard, widget, and admin module (e.g., RFM chart layout).

3.4 Shopify Integration: App Bridge

- *Why*: Embeds `React` components in Shopify's admin and storefront.
- *Use Case*: Integrates dashboard with Shopify admin and widget with storefront themes.

3.5 Visualization: Chart.js

- *Why*: Renders RFM segment bar charts in `AnalyticsPage.tsx` and admin module.
- *Use Case*: Displays RFM segments (e.g., "Best Customers: 10%") and Phase 2 analytics.

4 Deployment and Testing

4.1 Deployment: VPS (e.g., Ubuntu with Docker)

- *Why*: Replaces Railway for hosting `NestJS`, `PostgreSQL`, and `Redis`, giving full control over infrastructure. `Docker` simplifies deployment of backend (`NestJS`), databases

(PostgreSQL, Redis), and frontend (Vite + React) with consistent environments. Supports CI/CD via GitHub Actions.

- *Use Case*: Hosts TVP backend APIs, frontend assets, and databases. Scales to 5,000+ merchants with Redis caching and PostgreSQL indexes.
- *AI Assistance*: Provide Docker setup guides (e.g., "Dockerize NestJS and PostgreSQL for VPS") and Nginx configs for serving the frontend.

4.2 CI/CD: GitHub Actions

- *Why*: Automates testing and deployment to VPS, ensuring code quality.
- *Use Case*: Runs Jest/Cypress tests and deploys to VPS on git push.

4.3 Testing: Jest, Cypress, cargo test

- *Why*: Jest for unit/integration tests (NestJS APIs, RFM logic), Cypress for end-to-end tests (dashboard, widget, RFM UI), and cargo test for Rust Functions.
- *Use Case*: Tests APIs, frontend components, Shopify/Twilio integrations, and Rust logic. Load tests validate 5,000 customers.
- *AI Assistance*: Generate test cases (e.g., "Write Jest tests for RFM API").

5 Integrations

5.1 Shopify: @shopify/shopify-app-express (backend), Shopify CLI (Rust Functions)

- *Why*: Handles OAuth, webhooks (orders/create), and POS points earning (1 point/$).
- *Use Case*: Authenticates merchants, syncs orders, and enables POS integration.

5.2 Twilio

- *Why*: Powers SMS referrals in customer widget (e.g., 10% off).
- *Use Case*: Sends referral codes, tracked in referrals table.

6 Development Tools

6.1 AI Tools: GitHub Copilot, Cursor, Grok

- *Why*: Generates code, tests, and schema optimizations, with manual review for Shopify compliance.
- *Use Case*: Speeds up coding (e.g., NestJS APIs, React components) and learning (e.g., "Explain TypeScript interfaces").

6.2 Version Control: Git (GitHub)

- *Why*: Tracks changes, prevents loss, and supports CI/CD.
- *Use Case*: Commits code (e.g., git commit -m "Add points API").