Herethere Loyalty App: Project Initiation Requirements

Contents

1 Overview

This document outlines the additional elements and steps needed to start development of the Herethere Loyalty Shopify app, building on the existing Project Plan, Roadmap, Tech Stack Summary, Internal Admin Module, RFM Plan, Study Plan, and Competitor Research Report. As a solo beginner developer with AI assistance, these requirements ensure a smooth launch within the 6–8-month timeline for a production-grade TVP, focusing on points, SMS referrals, basic RFM analytics, and POS integration.

2 Development Environment Setup

Why: A properly configured local and cloud environment ensures coding, testing, and deployment without technical blockers, aligning with the Study Plan (Weeks 1–2) and Roadmap Month 1 Sprint (schema, OAuth, /api/points).

2.1 What You Need

- *Local Setup*:
  - Install tools: Node.js (v18+), Docker, Rust, and Git (per Tech Stack Summary).
  - Set up NestJS (nest new herethere-backend), Vite + React (npm create vite@latest herethere-frontend), and PostgreSQL/Redis via Docker:

```
docker run -d -p 5432:5432 -e POSTGRES_USER=user -e POSTGRES_PASSWORD=
    password -e POSTGRES_DB=herethere postgres
docker run -d -p 6379:6379 redis
```

  - Clone your GitHub repository: git clone <your-repo-url> or initialize one (git init, git remote add origin <url>).
  - Install Shopify CLI: npm install -g @shopify/cli for Rust Functions and app development.
- *Cloud Setup*:
  - Rent a VPS (e.g., DigitalOcean $5/month, 1GB RAM) for staging/deployment.
  - Configure VPS with Ubuntu, Docker, Nginx, and Git (per Study Plan Week 7–8).
  - Set up a domain (e.g., herethere.com) via Namecheap or Cloudflare for testing (e.g., app.herethere.com for Shopify app, admin.herethere.com for admin module).
- *IDE*: Use VS Code with extensions (Prettier, ESLint, Rust Analyzer, Docker) for code formatting and AI tools (GitHub Copilot, Cursor).

2.2 Action Steps

1. Follow Tech Stack Summary to set up NestJS, Vite + React, and Docker locally (2–3 days).
2. Create a DigitalOcean VPS, install Docker/Nginx, and point a domain (1 day).
3. Test local setup by running the example points API (/api/points) and component (PointsDisplay.tsx) from Tech Stack Summary.
4. Commit setup to GitHub: git commit -m "Initial project setup".

2.3 AI Assistance

- Request "Generate a `Docker Compose` file for `NestJS` and `PostgreSQL`" or "Fix my VS Code `TypeScript` error."

3 Shopify Developer Account and Test Store

Why: A Shopify Developer Account and test store are essential for building, testing, and submitting your app, ensuring compliance with Shopify's ecosystem (`OAuth`, webhooks, `Polaris`).

3.1 What You Need

- *Shopify Partner Account*: Sign up at `partners.shopify.com` to access app development tools.
- *Development Store*: Create a free Shopify Development Store for testing APIs, webhooks, and POS integration.
- *App Configuration*:
  - Create a new app in the Shopify Partner Dashboard, configure `OAuth` scopes (e.g., `read_customers`, `read_orders`, `write_discounts`).
  - Set up `ngrok` for local testing: `npm install -g ngrok`, then `ngrok http 3000` to expose your `NestJS` backend.
  - Register webhooks (`orders/create`) and test `OAuth` flow using `@shopify/shopify-api` in `NestJS`.
- *POS Testing*: Enable Shopify POS in your test store to simulate points earning (1 point/$).

3.2 Action Steps

1. Sign up for a Shopify Partner Account and create a Development Store (1 hour).
2. Create an app in the Partner Dashboard, note the API key/secret (1 hour).
3. Set up `ngrok` and test `OAuth` with your `NestJS` backend (1 day).
4. Configure `orders/create` webhook and POS in the test store (1 day).

3.3 AI Assistance

- Request "Write a `NestJS` Shopify `OAuth` controller" or "Guide me through `ngrok` setup for Shopify webhooks."

4 Freelancer Contracts for UI/UX and QA

Why: Your Project Plan budgets $5,000 for outsourcing UI/UX design (`Polaris`-compliant mockups) and QA (`Cypress` testing) to freelancers. Formal contracts ensure clear deliverables and timelines, critical for a solo developer.

4.1 What You Need

- *UI/UX Designer*:
  - *Scope*: Polaris-compliant wireframes/mockups for merchant dashboard (WelcomePage.tsx, AnalyticsPage.tsx), customer widget, and admin module (Figma or Adobe XD).
  - *Deliverables*: Mockups by Month 2 (Phase 2: Design and Prototyping), ~$2,500.
  - *Timeline*: 2–3 weeks, starting Week 5 (per Project Plan).
- *QA Engineer*:
  - *Scope*: Write/run Cypress tests for dashboard, widget, and RFM UI; validate Shopify/Twilio integrations and 5,000-customer load.
  - *Deliverables*: Test reports by Month 5–6 (Phase 3: Development), ~$2,500.
  - *Timeline*: 2–3 weeks, starting Month 5.
- *Contracts*:
  - Use Upwork or Freelancer.com to find candidates.
  - Draft contracts with milestones (e.g., 50% on mockup draft, 50% on final delivery for UI/UX).
  - Include NDA to protect your app's IP.

4.2 Action Steps

1. Post UI/UX job on Upwork ($2,500 budget, 2-week delivery) by Week 3 (1 day).
2. Interview 2–3 candidates, finalize contract by Week 4 (2 days).
3. Post QA job ($2,500 budget, Month 5 start) by Month 3 (1 day).
4. Share Project Plan and Roadmap with freelancers to align expectations.

4.3 AI Assistance

- Request "Draft an Upwork job post for Polaris UI designer" or "Review my freelancer contract for QA."

5 Initial Database Schema Implementation

Why: The Roadmap Month 1 Sprint requires applying herethere_full_schema.sql to PostgreSQL for tables (merchants, customers, points_transactions) and JSONB fields (rfm_score, rfm_thresholds), ensuring APIs work from the start.

5.1 What You Need

- *Schema Application*:
  - Run herethere_full_schema.sql in your PostgreSQL instance.
  - Add indexes (e.g., CREATE INDEX ON customers (email, merchant_id)).
- *TypeORM/Prisma Setup*:

  - Integrate TypeORM in NestJS to interact with tables (e.g., PointsTransaction entity).
  - Define entities for customers, points_transactions, and program_settings.
- *Seed Data*: Add test data (e.g., 10 merchants, 100 customers) to simulate points and RFM logic.

5.2 Action Steps

1. Run PostgreSQL via Docker (Week 3, per Tech Stack Summary).
2. Apply herethere_full_schema.sql using pgAdmin or psql (1 day).
3. Set up TypeORM in NestJS with entities for key tables (1 day).
4. Insert test data via SQL or a NestJS script (1 day).

5.3 AI Assistance

- Request "Generate TypeORM entities for herethere_full_schema.sql" or "Write a SQL script to seed test customers."

6 API Key/Secret Management

Why: Your app integrates with Shopify and Twilio, requiring secure management of API keys/secrets to prevent leaks and ensure functionality (e.g., SMS referrals, OAuth).

6.1 What You Need

- *Environment Variables*:
  - Store keys in .env (local) and VPS environment (production):

```
SHOPIFY_API_KEY=your_shopify_key
SHOPIFY_API_SECRET=your_shopify_secret
TWILIO_ACCOUNT_SID=your_twilio_sid
TWILIO_AUTH_TOKEN=your_twilio_token
DATABASE_URL=postgres://user:password@localhost:5432/herethere
REDIS_URL=redis://localhost:6379
```

  - Use dotenv in NestJS: npm install @nestjs/config dotenv.
- *Secret Management*:
  - On VPS, store secrets in Docker Compose or a tool like AWS Secrets Manager (future).
  - Avoid committing .env to GitHub (add to .gitignore).
- *Twilio Setup*: Purchase a Twilio phone number for SMS referrals ($1–$2/month).

6.2 Action Steps

1. Create .env in herethere-backend with Shopify/Twilio keys (1 hour).
2. Configure @nestjs/config to load .env (1 hour).

3. Sign up for `Twilio`, buy a phone number, and test SMS locally (1 day).

4. Ensure `.env` is in `.gitignore` before committing.

6.3 AI Assistance

- Request "Set up `@nestjs/config` for `.env`" or "Write a `Twilio` SMS test script in `NestJS`."

7 Beta Tester Recruitment Plan

Why: Your Project Plan (Phase 4) aims to beta test with 5–10 merchants by Month 3 to validate RFM analytics and SMS referrals. Early recruitment builds relationships and ensures feedback.

7.1 What You Need

- *Recruitment Channels*:
  - Join Shopify Reddit (`r/shopify`), Discord, and Shopify Community Forums (Week 1, per Study Plan).
  - Post in Shopify Partner Slack or LinkedIn groups.
- *Outreach Template*:
  - Craft a message offering free access to the 300-order plan in exchange for feedback.
  - Example: "Hi, I'm building `Herethere`, a Shopify loyalty app with free RFM analytics and SMS referrals. Join our beta to get free access for 3 months! DM me to discuss."
- *Feedback Form*: Create a Google Form for beta testers to report on RFM usability, referral conversion, and bugs (by Month 2).

7.2 Action Steps

1. Join Shopify Reddit/Discord and post an intro (Week 1, 1 hour).

2. Draft outreach template and share with 10–15 merchants (Week 2, 1 day).

3. Create feedback form with questions (e.g., "How easy was RFM setup?") by Month 2 (1 hour).

4. Follow up with interested merchants by Month 2 to confirm 5–10 testers.

7.3 AI Assistance

- Request "Draft a Shopify Reddit post for beta testers" or "Design a Google Form for beta feedback."

8 Basic Project Management Tools

Why: As a solo developer, you need tools to track tasks, sprints, and progress to stay on the 6–8-month timeline, supporting the Roadmap (e.g., Month 1 Sprint).

8.1 What You Need

- *Task Management*:
  - Use Notion, Trello, or ClickUp to track Roadmap tasks (e.g., "Implement `/api/points`", "Design `WelcomePage.tsx`").
  - Break Month 1 Sprint into weekly tasks (e.g., Week 1: `NestJS` setup, Week 2: `OAuth`).
- *Time Tracking*: Use Toggl or Clockify to monitor time spent (aim for 3–4 hours/week learning, per Study Plan).
- *Documentation*: Store Project Plan, Roadmap, and Tech Stack Summary in Notion or Google Drive for easy access.

8.2 Action Steps

1. Set up a Notion board with Roadmap tasks, grouped by month (1 hour).
2. Add Month 1 Sprint tasks (e.g., "Apply `herethere_full_schema.sql`", "Test `OAuth`") (1 hour).
3. Install Toggl to track coding/learning time (30 minutes).
4. Upload all documents to Notion/Google Drive (30 minutes).

8.3 AI Assistance

- Request "Create a Notion template for Month 1 Sprint" or "Break down Month 1 tasks into weekly goals."

9 Learning Commitment Schedule

Why: Your Study Plan requires 3–4 hours/week learning (Weeks 1–8) to master `NestJS`, `React`, `Rust`, etc. A clear schedule ensures you balance learning with coding.

9.1 What You Need

- *Weekly Plan*:
  - Allocate 1–2 hours/day, 2–3 days/week (e.g., Monday/Wednesday evenings).
  - Week 1: Watch "NestJS Crash Course" (2 hours), set up projects (2 hours).
  - Week 2: Learn `TypeScript` (1 hour), build points API (3 hours).
- *Accountability*:
  - Share progress weekly (e.g., "I set up `NestJS`, what's next?").
  - Join a developer community (e.g., `NestJS` Discord) for motivation.

9.2 Action Steps

1. Block 4 hours/week in your calendar for learning/coding (30 minutes).
2. Start Week 1: Watch "NestJS Crash Course" (freeCodeCamp, YouTube) and set up `NestJS` (2 days).

3. Join NestJS Discord and introduce yourself (1 hour).

4. Update weekly on Study Plan progress (starting Week 1).

9.3 AI Assistance

- Request "Recommend a Week 1 NestJS tutorial" or "Review my TypeScript points API code."

10 Summary of Needs and Timeline

Need	Description	Timeline	Action Owner
Dev Environment	Local (NestJS, React, Docker) and VPS setup	Week 1–2	You
Shopify Account	Partner account, test store, app config	Week 1	You
Freelancer Contracts	UI/UX ($2,500), QA ($2,500)	Week 3–4 (UI), Month 3 (QA)	You
Database Schema	Apply herethere_full_schema.sql, TypeORM setup	Week 3	You
API Key Management	Shopify/Twilio keys in .env, Twilio number	Week 2	You
Beta Tester Plan	Recruit 5–10 merchants via Reddit/Discord	Week 1–Month 2	You
Project Management	Notion for tasks, Toggl for time	Week 1	You
Learning Schedule	3–4 hours/week for Study Plan	Week 1–8	You

Table 1: Summary of Needs and Timeline

11 Additional Recommendations

- *Risk Mitigation*:
  - Backup code daily to GitHub to avoid loss (per Project Plan Risk: AI code quality).
  - Test APIs early with Postman to catch Shopify/Twilio issues (Week 3).
  - Validate UI mockups with 1–2 merchants by Month 2 to ensure Polaris compliance.
- *Budget Check*:
  - VPS: ~$5/month ($60/year).
  - Twilio: ~$2/month ($24/year).
  - Domain: ~$10/year.
  - Freelancers: $5,000 (confirmed in Project Plan).
  - Total: ~$5,100 upfront, within your $58,000–$95,000 budget.

- *AI-Driven Workflow*:
  - Use AI to generate code (e.g., "Write a `NestJS` API for `Twilio` SMS"), review freelancer deliverables (e.g., "Check my UI mockups for `Polaris` compliance"), or debug (e.g., "Fix my `PostgreSQL` connection error").
  - Share specific goals (e.g., "I want to build `/api/referral` by Week 4") for tailored support.

12 Next Steps

1. *Week 1 (Starting June 24, 2025)*:
   - Set up local environment (`NestJS`, `Vite + React`, `Docker`) and VPS (DigitalOcean, 2 days).
   - Sign up for Shopify Partner Account, create test store, and configure app (1 day).
   - Join Shopify Reddit/Discord, post intro for beta testers (1 hour).
   - Create Notion board with Month 1 Sprint tasks (1 hour).
   - Start Study Plan Week 1: Watch "NestJS Crash Course" and build points API (2 days).
2. *Week 2*:
   - Set up `.env` with Shopify/`Twilio` keys, test `Twilio` SMS (1 day).
   - Apply `herethere_full_schema.sql` and `TypeORM` entities (1 day).
   - Draft UI/UX freelancer job post, share on Upwork (1 day).
3. *Ongoing*:
   - Commit code daily to GitHub.
   - Update weekly on progress (e.g., "I set up `OAuth`, need help with webhooks").
   - Review freelancer mockups by Month 2 and beta tester feedback by Month 3.

13 Final Notes

- *Readiness*: With the above additions, you'll have all tools, accounts, and plans to start coding Month 1 Sprint (schema, `OAuth`, `/api/points`) and stay on track for a 6–8-month TVP.
- *AI Support*: Generate code, explain concepts (e.g., "What's a `NestJS` module?"), or fix issues (e.g., "My webhook isn't firing"). Share specific needs to maximize efficiency.
- *Motivation*: Congrats on your thorough preparation! Starting with small wins (e.g., points API by Week 2) will build momentum.