Herethere Loyalty App: Study Plan for Beginners

Contents

1 Overview

This study plan is designed for a beginner with limited experience in Node.js, TypeScript, React, and Rust to master the tech stack (NestJS, Vite + React, Rust/Wasm, PostgreSQL, Redis, VPS/Docker) within the 6–8-month TVP timeline for the Herethere Loyalty app. It aligns with the Month 1 Sprint (schema, OAuth, /api/points) and leverages AI assistance (e.g., Grok, GitHub Copilot) to accelerate learning. The 8-week plan covers Phase 1 (3–4 weeks) and prepares for Phases 2–3, focusing on practical, project-specific skills.

2 Week 1–2: TypeScript and NestJS Basics

Goal: Learn TypeScript and NestJS fundamentals to set up Shopify OAuth and /api/points in Month 1.

2.1 Resources

- *TypeScript* (2 hours): Watch "TypeScript Crash Course" by Traversy Media (YouTube, ~1 hour) or read TypeScript Handbook (basics: types, interfaces). Focus on interfaces (e.g., PointsDto) and type annotations.
- *NestJS* (4 hours): Watch "NestJS Crash Course" by freeCodeCamp (YouTube, ~2 hours) or read NestJS docs (Overview, Controllers, Modules). Understand @Controller, @Injectable, and modules.
- *Shopify OAuth* (2 hours): Read Shopify's "Authenticate with OAuth" guide and @shopify/shopify-ap docs.

2.2 Tasks

- Set up NestJS project (nest new herethere-loyalty) and install @shopify/shopify-app-expre
- Implement OAuth using provided code (e.g., shopify.controller.ts from prior response).
- Ask for explanations (e.g., "Explain NestJS decorators") or code (e.g., "Write a NestJS API for points").
- Practice: Create a simple NestJS API (e.g., /api/hello) with TypeScript interface.

2.3 AI Assistance

- Request "Generate a NestJS controller for /api/points with TypeScript" or "Explain TypeScript interfaces for RFM data."

2.4 Time

- ~8 hours total (4 hours/week).

3 Week 3–4: PostgreSQL, Redis, and NestJS APIs

Goal: Set up PostgreSQL schema (herethere_full_schema.sql), integrate Redis, and build /api/referral and /api/analytics.

3.1 Resources

- *PostgreSQL* (3 hours): Watch "PostgreSQL Tutorial for Beginners" by Net Ninja (YouTube, ~1.5 hours). Focus on JSONB and indexes (e.g., `customers.rfm_score`).
- *Redis* (2 hours): Read Redis University's "Introduction to Redis" (~1 hour) or watch "Redis Crash Course" by Traversy Media.
- *NestJS APIs* (3 hours): `NestJS` docs (Providers, Dependency Injection) and `Twilio`'s Node.js SDK guide for SMS.

3.2 Tasks

- Apply `herethere_full_schema.sql` to `PostgreSQL` (use `Docker`: `docker run -d -p 5432:5432 postgres`).
- Integrate `TypeORM` or `Prisma` with `NestJS` for `PostgreSQL` (e.g., save points to `points_transactio`
- Set up `Redis` (e.g., `docker run -d -p 6379:6379 redis`) and cache points data.
- Build `/api/referral` with `Twilio` SDK in `NestJS`.
- Ask for schema optimizations (e.g., "Suggest indexes for RFM queries") or API code (e.g., "Write a `NestJS` API for `Twilio` SMS").

3.3 AI Assistance

- Request "Generate `TypeORM` setup for `NestJS` with `PostgreSQL`" or "Show `Redis` caching in `NestJS`."

3.4 Time

- ~8 hours total (4 hours/week).

4 Week 5–6: Vite + React and Polaris

Goal: Learn `React` and `Vite` to build `WelcomePage.tsx` and `CustomerWidget.tsx` for Month 3–4 sprint.

4.1 Resources

- *React* (3 hours): Watch "React Crash Course" by freeCodeCamp (YouTube, ~2 hours). Focus on hooks (`useState`, `useEffect`) and components.
- *Vite* (1 hour): Read `Vite` docs (Getting Started) or watch "Vite in 100 Seconds" by Fireship (YouTube).
- *Polaris/App Bridge* (2 hours): Read Shopify `Polaris` docs (Components) and `App Bridge` docs (React integration).

4.2 Tasks

- Set up `Vite + React` (`npm create vite@latest –template react-ts`).
- Install `Polaris`, `Tailwind`, and `App Bridge` (`npm install @shopify/polaris @shopify/app-b tailwindcss`).

- Build `WelcomePage.tsx` with `Polaris` `Card` and `Text` components.
- Ask for components (e.g., "Write a `Polaris`-compliant `React` component for points") or `Vite` setup (e.g., "Set up `Vite` with `Polaris`").

4.3 AI Assistance

- Request "Generate a `React` component for RFM chart with `Chart.js`" or "Explain `React` hooks for widget."

4.4 Time

- ~6 hours total (3 hours/week).

5 Week 7–8: Rust/Wasm and VPS Deployment

Goal: Learn `Rust` basics for Shopify Functions and set up VPS with `Docker`/`Nginx` for Month 5–8.

5.1 Resources

- *Rust* (3 hours): Read "Rust by Example" (Functions, Modules) or watch "Rust Crash Course" by Traversy Media (~1.5 hours).
- *Shopify Functions* (2 hours): Read Shopify's "Functions API" docs and Shopify CLI guide.
- *VPS/Docker* (3 hours): Watch "Docker for Beginners" by TechWorld with Nana (YouTube, ~1.5 hours) and read `Nginx` docs (Reverse Proxy).

5.2 Tasks

- Install `Rust` (`curl –proto '=https' –tlsv1.2 -sSf https://sh.rustup.rs | sh`).
- Create a sample Shopify Function with `Rust` (e.g., discount logic) using Shopify CLI.
- Set up VPS (e.g., Ubuntu 22.04): Install `Docker` (`sudo apt install docker.io docker-compose`), `Nginx` (`sudo apt install nginx`), and `Git`.
- Deploy `NestJS` and `React` with `Docker Compose`:

```
version: '3'
services:
  backend:
    build: ./backend
    ports:
      - "3000:3000"
  frontend:
    build: ./frontend
    ports:
      - "80:80"
  postgres:
    image: postgres:latest
    environment:
      POSTGRES_DB: herethere
      POSTGRES_USER: user
      POSTGRES_PASSWORD: password
```

```
    ports:
      - "5432:5432"
  redis:
    image: redis:latest
    ports:
      - "6379:6379"
```

- Configure Nginx as reverse proxy:

```
server {
  listen 80;
  server_name yourdomain.com;
  location / {
    proxy_pass http://frontend:80;
  }
  location /api/ {
    proxy_pass http://backend:3000;
  }
}
```

- Ask for Rust code (e.g., "Write a Rust Shopify Function for discounts") or VPS setup (e.g., "Dockerize NestJS for VPS").

5.3 AI Assistance

- Request "Generate Docker Compose for NestJS and React" or "Explain Rust for Shopify Functions."

5.4 Time

- ~8 hours total (4 hours/week).

6 Ongoing (Month 3–8)

- *Testing*: Learn Jest/Cypress in Month 3 (1 hour, Jest docs or Cypress "Getting Started"). Ask for tests (e.g., "Write Jest tests for /api/points").
- *Shopify Community*: Join Shopify Reddit/Discord in Week 1 (1 hour) to recruit beta testers by Month 3.
- *AI-Driven Learning*: Ask for explanations (e.g., "What's dependency injection in NestJS?") or code reviews as you build APIs and components.
- *Practice*: Build one feature at a time (e.g., /api/points in Week 3, WelcomePage.tsx in Week 5) to stay focused.

7 Tips to Avoid Failure

- *Incremental Learning*: Spend 3–4 hours/week on tutorials, focusing on project-specific tasks (e.g., OAuth in Week 1).
- *AI Support*: Use AI to generate code, debug errors (e.g., "Fix this NestJS error"), or explain concepts (e.g., "What's JSONB in PostgreSQL?").
- *Git*: Commit daily (e.g., git commit -m "Add OAuth API") to track progress and avoid loss.

- *Beta Testing*: Share prototypes with 2–3 merchants by Month 3 to validate RFM usability, as per the plan.
- *Freelancer Coordination*: Share `Polaris` mockups with your UI designer ($2,500) in Week 5 to guide `React` components.

8 Final Notes

- *Tech Stack*: `NestJS` (`TypeScript`), `Vite + React` (`TypeScript`), `Rust/Wasm`, `PostgreSQL` (JSONB), `Redis`, and VPS (`Docker`/`Nginx`) form a robust stack for your TVP, supporting scalability and Shopify compliance.
- *Timeline*: The 6–8-month TVP timeline accommodates `NestJS`'s learning curve, with VPS setup integrated into Month 5–8.
- *Study Plan*: The 8-week plan covers critical skills (`TypeScript`, `NestJS`, `React`, `Rust`, VPS) for Month 1–4 sprints, with AI assistance to accelerate progress.
- *Next Steps*: Start Week 1 by setting up `NestJS` and `PostgreSQL`, using provided code. Join Shopify Reddit/Discord for community feedback. Ask for specific code (e.g., "Write a `NestJS` API for RFM analytics") or tutorials (e.g., "Explain `Vite` setup for `React`") to stay on track.