Parcel Tracker App

Build a parcel tracking web app. Users track shipments, manage history, and share results.

Stack: Next.js (App Router), React, Supabase, Tailwind CSS
External API: TrackingMore API
Deployment: Should work locally and on Vercel

Requirements

Authentication

Users can sign up and log in with email/password via Supabase Auth. There are three levels of access:

- Guests can track a single parcel without an account
- Registered users can track parcels, save them, and manage their history
- Admins can see all users' tracking activity and usage statistics

Access control should be enforced at the database level.

Tracking

When a user enters a tracking number, the app detects the courier automatically, lets the user confirm or change it, then shows the tracking result with a status timeline and delivery checkpoints.

Users should be able to track up to 6 parcels at once. The form supports adding, removing, and reordering tracking fields via drag and drop.

History

Logged-in users have all their trackings saved. The history page lets them:

- See all past trackings with their current status
- Refresh any tracking to pull the latest data
- Delete one or many trackings at once

- Filter by status and sort by date

Live Updates

If a user refreshes a tracking from one device, any other open tab or device logged into the same account should reflect the updated status immediately — without a manual page reload.

Dashboard

Logged-in users see a dashboard with:

- Summary cards (total trackings, in transit, delivered this month, average delivery time)
- A chart of tracking volume over the past 30 days
- Courier breakdown showing usage frequency and delivery success rates
- A list of currently in-transit parcels with quick-refresh

All statistics should be computed server-side.

Bulk Import

Users can upload a CSV file with tracking numbers and optional courier codes to create many trackings at once.

After upload, show a validation report. Let the user review and confirm before submitting. Show progress during submission and a summary when done.

Should handle large files smoothly.

Shareable Tracking Pages

Each tracking gets a unique public URL that shows the full timeline without requiring login. The page should include a QR code and generate an OG image for social media previews showing the tracking status.

Owners can toggle any tracking between public and private.

Contact Form

A contact page that sends an email to a configured address. Validate inputs server-side. Limit how often the same visitor can submit.

Testing

Write Playwright end-to-end tests for:

- Auth flow (sign up, log in, log out, protected routes)
- Tracking as guest and as logged-in user
- History verification after tracking
- Bulk CSV import with mixed valid/invalid data
- Public/private toggle on shared tracking pages

Tests must run locally and reliably, independent of external services.

UI Expectations

- Responsive across mobile, tablet, desktop
- Dark and light mode
- Loading states and skeletons
- Animations for transitions and list interactions
- Toast notifications for actions and errors
- Empty states where appropriate

Constraints

- No secrets exposed to the client
- External API calls must go through your own API routes
- TypeScript throughout
- Database tables need foreign keys, indexes, and row-level security
- No hardcoded data

Development Environment

You must use **Claude Code** for development. An API key will be provided by the interviewer.

Install and configure **Entire CLI** (https://github.com/entireio/cli) before you start coding. Entire CLI records your Claude Code sessions and pushes them to GitHub — this is how we review your development process, not just the final code.

Make sure Entire CLI is working and syncing sessions before you begin. Verify that sessions are appearing on your GitHub. If it's not pushing, debug it before moving on.

Workflow

- Work in a **public GitHub repo**
- Make **multiple meaningful commits** as you go — don't build everything and commit once at the end
- Commits should reflect your actual development progression (e.g., auth setup, then tracking, then history, etc.)
- We review both your code and your Claude Code session history via Entire CLI

Setup

You'll need:

1. A Supabase project
2. A TrackingMore account
3. An SMTP provider for the contact form
4. Claude Code with the API key provided to you
5. Entire CLI installed and configured (https://github.com/entireio/cli)

Deliverable

- A **public GitHub repo** that can be cloned, configured with `.env.local`, and run locally
- Should also deploy to Vercel
- Include a README explaining setup and any decisions you made
- Entire CLI sessions must be visible on the repo — this is part of the submission