ChatGPT

Metaprompt for RouteOptimizer Frontend

Project Context: RouteOptimizer is a real-time delivery routing system built with React and TypeScript [1]. The new frontend must add a unified analytics layer with role-based dashboards (Vehicle Owner, Customer, etc.), integrating geospatial and environmental data sources.

Cross-Perspective Analytics Framework

- **Real-Time Pipeline:** Use WebSockets or similar for live updates and a unified event tracking system.
- **Modular Charts:** Build reusable chart components (e.g. with D3.js or Chart.js) and template layouts for each dashboard perspective.

Vehicle Owner Dashboard

- **Core Metrics:** Fleet utilization, maintenance cost trends, fuel efficiency.
- **Route Performance (GraphHopper):** Use GraphHopper's Directions API for optimized routing and efficiency scoring (GraphHopper enables fast route planning and up to 30% time/fuel savings [2]). Show historical route efficiency and idling-time heatmaps.
- **Environmental Impact (Ambee + OpenWeather):** Integrate Ambee's air-quality/emissions data and OpenWeather forecasts. For example, use Ambee's environmental intelligence for emissions tracking and risk assessment [3], and use OpenWeather's API to get current/forecast weather data [4] (e.g. adjust ETA by +20% when rain >10%).
- **Profitability:** Calculate revenue per mile/km and cost breakdown by vehicle.

Customer Dashboard

- **Core Metrics:** Delivery reliability (on-time %), satisfaction trends, price sensitivity.
- **Route Transparency (Mapbox):** Use Mapbox's mapping and navigation APIs for live tracking and ETA accuracy [5]. Show live ETA vs actual performance and route deviation alerts.
- **Service Conditions (OpenWeather/Ambee):** Display weather impacts (from OpenWeather) and air quality (from Ambee) during deliveries [4] [3] (e.g., alert when poor air quality).
- **Comparative Analytics:** Benchmark different providers or service tiers (e.g., compare on-time rates and costs).

Route Optimization Hub (Multi-View)

- **Operator View:** Show GraphHopper-based routing costs and overall optimization performance [2].
- **Driver View:** Display Mapbox-driven navigation analytics (e.g. deviation from planned route, navigation time).
- **Customer View:** Use MapMyIndia's routing/ETA accuracy for end-user perspective (MapMyIndia offers global mapping, routing, and search APIs [6]).

API Integration Matrix

Use the following APIs for specific use cases:
- **GraphHopper:** Route optimization and cost estimates [2] (e.g. maintenance route planning, delivery cost per route).
- **Mapbox:** Real-time tracking and navigation [5] (e.g. live position updates, map visualization).
- **Ambee:** Environmental monitoring [3] (e.g. cabin air quality, health alerts during delivery).
- **OpenWeather:** Weather forecasting [4] (e.g. fleet weather risk, delay prediction).
- **MapMyIndia:** Regional maps and geocoding [6] (e.g. localized routing compliance, address validation).

Implementation Requirements

- **Role-Based Rendering:** Filter which metrics each user sees (Owner/Driver/Customer) and support unit toggling (km/miles, ₹/$).
- **Data Fusion:** Combine data from APIs. *Example:* `eta = base_eta * (weather['precipitation']>10 ? 1.2 : 1.0)`.
- **Alerts:** Configure threshold alerts: maintenance reminders for owners, delay notifications for customers, etc.

Project Setup (shadcn/UI, Tailwind, TypeScript)

- The project should use shadcn/ui conventions (React+TS+Tailwind). The default components folder is `/components/ui` [7]; create it if it doesn't exist.
- **Initialize shadcn UI:** Run `npx shadcn@latest init` to scaffold the project structure [8].
- **Tailwind CSS:** Install and configure Tailwind v4 (`npm install tailwindcss@latest`) [9]. Ensure `tailwind.config.js` and `styles/globals.css` are set up.
- **TypeScript:** Install TS and typings (`npm install typescript @types/react @types/node --save-dev`) [10] and run `npx tsc --init`.
- **Path Aliases:** In `tsconfig.json`, set `"baseUrl": "."` and `"paths": { "@/*": ["./*"] }` to allow `@/` imports [11].
- **Dependencies:** Add required libs: `npm install lucide-react` [12] (Shadcn UI uses `lucide-react` for icons), plus `clsx`, `tailwind-merge`, etc., as per shadcn documentation [12].

HeroSection Component Integration

- Create the file `/components/ui/hero-section-dark.tsx` and paste the provided `HeroSection` code into it. Also create a demo page/component (e.g. `HeroSectionDemo`) to import and render it.
- Update asset URLs: replace `bottomImage.light` and `.dark` with valid image links (e.g. Unsplash URLs).
- Ensure the `ChevronRight` icon is imported from `lucide-react`.
- The component uses Tailwind classes (including dark-mode styles and animations) and CSS variables for the retro grid; verify these styles work with the Tailwind setup.

Integration Guidelines

1. **Analyze the Component:** Identify all props and dependencies. The `HeroSection` takes `title`, `subtitle`, `description`, `ctaText`, `ctaHref`, `bottomImage`, and `gridOptions` props and spreads extra `className`. It uses a forwarded ref (`React.forwardRef`).
2. **Dependencies:** It imports `cn` (a classnames utility) and `ChevronRight` from `lucide-react`. Make sure `@/lib/utils` (for `cn`) exists and lucide-react is installed.
3. **State/Context:** The component has no internal state. It relies on parent-provided props. Ensure any theming context (e.g. dark mode provider) is present so `dark:` styles apply.
4. **Questions to Answer:** Determine how data flows to this component: Where do `title` and `subtitle` values come from? Is any global state or context needed? What images should be used for `bottomImage`? How should it behave on different screen sizes? Where in the app should this Hero section be placed (e.g. homepage)?
5. **Responsive Behavior:** The layout should adapt to mobile. The Tailwind classes (`max-w-screen-xl`, `mx-auto`, etc.) suggest responsiveness. Test on small screens.
6. **Final Placement:** Decide the best place for this component (e.g. a landing page or dashboard header) based on its content and style.

Integration Steps

1. **Copy Code:** Paste the provided `hero-section-dark.tsx` into `/components/ui/` and the demo code (`HeroSectionDemo`) into an appropriate file (e.g. `/pages` or `/app`).
2. **Install Dependencies:** Run `npm install lucide-react` [12] (and other shadcn dependencies if needed).
3. **Update Assets:** Replace placeholder image URLs with real ones (e.g. known Unsplash images).
4. **Verify Icons:** Ensure `ChevronRight` from `lucide-react` is rendering (no missing icons).
5. **Check Styling:** Make sure Tailwind classes are applied correctly (dark mode gradients, hover effects). Adjust CSS variables if needed.
6. **Test Component:** Render `HeroSectionDemo` to verify it appears as expected (text, styles, layout) in both light and dark mode, and on mobile.
7. **Iterate:** Fix any styling or import issues (e.g. if `@/components/ui` path needs adjusting in `tsconfig.json` aliases).

Finally, the metaprompt should explicitly instruct Gemini CLI to verify each of these points and build the frontend accordingly. It can also ask the user whether to include additional details (e.g. metric formulas, sample API schemas, mobile-optimization, anomaly detection) as needed for completeness.

Sources: The project's tech stack (React/TS) [1] and features like GraphHopper routing [2], Mapbox mapping [5], Ambee environmental data [3], OpenWeather forecasts [4], and MapMyIndia APIs [6] informed these instructions. Shadcn/UI installation and path conventions [8] [7] [12] were used for project setup guidance.

---

[1] GitHub - Manishak798/RouteOptimizer: RouteOptimizer is an innovative real-time delivery route optimization system.
https://github.com/Manishak798/RouteOptimizer

2 GraphHopper Directions API with Route Optimization
https://www.graphhopper.com/

3 Ambee
https://www.getambee.com/

4 Weather API - OpenWeatherMap
https://openweathermap.org/api

5 Saaspo | Mapbox Landing Page
https://saaspo.com/pages/mapbox-landing-page

6 MapmyIndia's Global APIs & SDKs | Explore the largest directory of APIs & SDKs
https://www.mapmyindia.com/api/global-api/

7 8 9 10 React component integration - v0 by Vercel
https://v0.dev/chat/react-component-integration-VxkywXia490

11 12 Manual Installation - shadcn/ui
https://ui.shadcn.com/docs/installation/manual