A Simple Guide to Our Latency Visualizer App

Hello! This guide explains how we built our Latency Topology Visualizer app. We wrote this in simple English so everyone can understand how it works.

# How the App is Organized

Think of our app's code like a house with different rooms. Every file has its place, which helps us keep things tidy.

* **App Folder:** This is the main entrance. It has the main page you see and the basic styles (colors and fonts).
* **Components Folder:** This is like a box of LEGO bricks. It holds all the small, reusable parts of our app, like the 3D globe, buttons, and menus.
* **Hooks Folder:** This folder has special tools that help our components do their jobs, like getting new data.
* **Lib Folder:** This is our toolbox. It has helpers for measuring speed and talking to the internet.
* **Data Folder:** This is where we keep our example data. This data helps us show you the map even before we get live information.
* **Types Folder:** This is our rulebook. It helps us make sure all our data is in the correct format, which prevents mistakes.

We used a few smart ideas to build the app:

* **One Main Tool:** We have one main tool that checks the connection speed for the whole app. This is better than having many tools doing the same thing.
* **Sharing News:** When new speed data arrives, this main tool tells all the different parts of the app that need to know.
* **LEGO Bricks:** We build big things from small parts. This makes it easy to fix or change things without breaking the whole app.

# The Main Parts of the App

These are the biggest and most important pieces you see and use.

## The 3D Earth Globe (MapboxGlobe.tsx)

* It looks like a real planet.
* It draws animated lines between places to show their connection speed.
* You can click on the dots (markers) to get more information.
* It can change between a light and dark look to match your screen.

## The Swipe-Up Menu (MobileControlPanel.tsx)

* If you are on a phone, you can swipe up from the bottom to open a special menu.
* It feels just like the control center on an iPhone.
* It has buttons to control the app.
* It uses smooth, bouncy animations when it opens and closes.

## The History Chart (HistoricalChart.tsx)

* This part shows you graphs of the connection speeds from the past.
* You can see how the speed changed over the last hour, day, or week.
* You can choose different cryptocurrencies to see their data.
* It shows you the highest, lowest, and average speed in that time.

## The Network Map (NetworkTopology.tsx)

* This shows you the path that data takes from one point to another.
* You can search for a specific location.
* It gives details like speed, connection quality, and the number of "hops" (stops) the data makes.

# Our Special Tools (Custom Hooks)

* **useRealTimeLatency.ts:** This tool is always working in the background. Every few seconds, it checks for new speed data and updates the screen. It also calculates the average, highest, and lowest speeds you see.
* **useStore.ts:** This is the app's memory. It remembers which locations you have clicked on and what filters you have set. We use a simple tool called Zustand for this.
* **useTheme.ts:** This tool handles the app's look. It can automatically switch between light and dark mode based on your phone or computer's system settings.

# Getting Information (API Classes)

* **Latency Monitor (LatencyMonitor):** This is our main speed checker. It sends a tiny, quick request to different places to measure how long it takes to get a response. If it can't get a real measurement, it uses our example data so the app keeps working.
* **Network Performance Monitor (NetworkPerformanceMonitor):** This helper checks how good your internet connection is. It looks at your network type (like 4G or WiFi) and your round-trip time (RTT).

# Making the App Fast

* Only Update What's Needed: We tell React (the library we use) not to rebuild parts of the screen that haven't changed. This saves a lot of work for your device.
* Reuse Map Markers: When the data on the map updates, we try to reuse the existing dots (markers) instead of creating new ones every single time. This makes the map feel much smoother.
* Smart Animations: We make sure that all animations are efficient and stop correctly when they are no longer needed, which saves battery.

# What Happens When Things Go Wrong

* API Errors: If the app can't get live data, it won't crash. Instead, it will show you a friendly error message and might use older data or our example data to keep the map working.
* Component Errors: If one small part of the app breaks, we show a "Something went wrong" message for just that part. This way, the rest of the app can still work fine.
* Trying Again: If a request to the internet fails, the app will wait a second and then try again automatically. It will try a few times before it gives up and shows an error.

# Keeping the App Safe and Working

* Testing: We test every part of our app to make sure it works correctly. We test individual "LEGO bricks" (components) and the special tools (hooks) to catch bugs early.
* Security: We are careful to keep the app secure. For example, we check any text that is typed into a search box to make sure it is safe.