## **Assignment: CryptoWeather Nexus**

**Goal:** Build and publicly deploy a modern, multi-page dashboard combining weather data, cryptocurrency information, and real-time notifications via WebSocket.

### **1. Scope and Timelines**

* **Timeframe:** This assignment is expected to be completed within **2 days**.
* **Deliverables:** Email to gaurav@userology.co  
  + A fully functional, **publicly deployed** web application.
  + A **GitHub repository** with clear commit history.
  + A **README** documenting setup, usage, and design decisions.

**Recommendation:** Feel free to use AI tools and LLMs (e.g., ChatGPT, Claude, Gemini, Copilot) or any online resources. However, be prepared to **think critically** and solve inevitable integration challenges.

### **2. Core Requirements**

1. **Framework & Libraries**
   * **Next.js (v13+)** with file-based routing.
   * **React** (hooks for state and lifecycle).
   * **Redux** (with async middleware like Redux Thunk or Saga).
   * **Tailwind CSS** for styling.
2. **Multi-Page Architecture**
   * **Dashboard** page with three sections:  
     + **Weather:** Show temperature, humidity, and conditions for at least **three** predefined cities (e.g., New York, London, Tokyo).
     + **Cryptocurrency:** Display live price, 24h change, and market cap for at least **three** cryptos (e.g., Bitcoin, Ethereum, and one more).
     + **News:** Show the **top five** crypto-related headlines.
   * **Detail Pages:**
     + **City** details (weather history, chart/table).
     + **Crypto** details (historical pricing, extended metrics).
3. **API Integrations**
   * **Weather Data:** OpenWeatherMap or any equivalent free API.
   * **Crypto Data:** CoinGecko, CoinCap, or any reputable free API.
   * **News Headlines:** NewsData.io or a similar free news API.
   * **Real-Time Data (WebSocket):**
     + Use CoinCap WebSocket for live price updates.  
       - Docs: [CoinCap WebSocket Documentation](https://docs.coincap.io/)
     + Simulate weather alerts by dispatching in-app WebSocket or similar mock events.
4. **Redux & State Management**
   * Store user preferences (favorite cities/cryptos) and all fetched data globally.
   * Implement loading/error states for robust UI feedback.
5. **Real-Time Notifications**
   * Establish a WebSocket connection to receive **price changes** for BTC/ETH.
   * Display notifications (toast or dropdown) for significant price shifts or simulated weather alerts.
   * Include a “type” field (e.g., price\_alert / weather\_alert) in payloads.
6. **Responsive UI & Basic Design System**
   * Use Tailwind CSS for layout, typography, and components.
   * Ensure consistent typography, spacing, and color usage.
   * **Responsive Layout:** Must adapt seamlessly from mobile screens to large desktops.
   * Interactive elements (buttons, links, toasts) should have clear hover, active, and focus states.
7. **Deployment**
   * Deploy on a public platform (e.g., [Vercel](https://vercel.com/) or [Netlify](https://www.netlify.com/)).
   * Manage **API keys** securely (e.g., environment variables).

### **3. Additional Nuances**

* **Data Refresh & Partial Failures:**
  + Periodically sync data (e.g., every 60s).
  + Handle partial outages gracefully and display fallback UI if an API call fails.
* **Favorites Feature:**
  + Let users “favorite” a city or crypto.
  + Persist and display these favorites in a special section or highlight them visually.
* **Routing Nuance:**
  + Handle deep links (e.g., /crypto/bitcoin) to ensure SSR/SSG data pre-fetching works.
* **Testing:**
  + (Optional) Provide unit tests for critical Redux logic or WebSocket message handling.

### **4. Submission Guidelines**

1. **GitHub Repository**
   * Public repo with commit history.
   * Include a **README** detailing setup, build instructions, and design decisions.
2. **Publicly Deployed Link**
   * Provide the live URL (e.g., https://yourapp.vercel.app).
3. **Documentation**
   * Summarize challenges and how they were resolved.
   * List any **alternative APIs** used if the suggested ones are unavailable.

### **5. Resources & Links**

* **Next.js Documentation:**<https://nextjs.org/docs>
* **Tailwind CSS Docs:**<https://tailwindcss.com/docs>
* **CoinCap WebSocket Docs:**<https://docs.coincap.io/>
* **OpenWeatherMap API:**<https://openweathermap.org/api>
* **CoinGecko API:**<https://www.coingecko.com/en/api>
* **NewsData.io:**<https://newsdata.io/>