Home Assignment - Full-Stack Developer (React + Node.js)

As the next step in our interview process, we would like you to complete a short home assignment. This task will help us evaluate your full-stack development skills using **React** and **Node.js**.

Assignment: Threat Intelligence Dashboard

Your goal is to build a simple full-stack application that allows users to input an IP address and retrieve threat intelligence data about it. The application should consist of a **React frontend** and a **Node.js backend for frontend (BFF)**.

React Frontend

Develop a user interface with the following features:

- **Input Field:** For entering an IP address.
- **"Check" Button:** To send the IP address to your backend for processing.
- **Results View:** Display the following information returned from the backend:
    - Threat/reputation score
    - Abuse reports
    - IP geolocation or similar metadata
- **State Handling:** Implement robust handling for loading and error states.
- **State Management:** Choose and implement a state management approach (e.g., Context API, Redux, etc).

Once a user enters a valid IP address and clicks “Check”, the app should display the following data fields: Upon a user's input of a valid IP address and subsequent click on "Check," the application is to present the following data fields:

Field	**Description**
IP address	The input IP (as validated by the backend)
Hostname (if available)	The hostname of the IP, if available
ISP	Internet service provider/carrier name
Country	Geolocation result (country name)

Abuse Score	A value from AbuseIPDB (0-100) indicates the likelihood of malicious use.
Recent Reports	How many abuse reports were submitted in the last X days.
VPN/Proxy Detected	Whether this IP is flagged as a VPN/proxy (from IPQualityScore).
Threat Score	An overall threat score, if available (from IPQualityScore or derived).

Node.js Backend-for-Frontend (BFF)

Create a BFF that acts as an intermediary between your frontend and external APIs:

- **API Route:** Expose an API endpoint (e.g., `/api/intel?ip=8.8.8.8`) that your frontend will consume.
- **IP Validation:** Implement validation to ensure the submitted IP address is in the correct format.
- **External API Integration:** Query at least two public APIs (suggestions below).
- **Data Aggregation:** Aggregate the responses from the external APIs into a clean, unified JSON format.
- **Response:** Return the aggregated result to the frontend.

Public API Suggestions

You may use any of the following free APIs or similar alternatives that serve the same purpose:

- **AbuseIPDB:** For IP abuse reports (requires a free API key).
    - Docs: https://www.abuseipdb.com/api.html
    - Sample call:

```
GET
https://api.abuseipdb.com/api/v2/check?ipAddress=8.8.8.8&maxAgeInDays=90
Headers:
  Key: Accept      → application/json
  Key: Key         → [Your API Key]
```

- **IPQualityScore:** For reputation and VPN detection (requires a free API key).
    - Docs: https://www.ipqualityscore.com/documentation/ip-reputation-api/overview
    - Sample call:

```
GET https://ipqualityscore.com/api/json/ip/[API_KEY]/8.8.8.8
```

API Keys

Feel free to use any of these API keys for your solution or register.You are welcome to use any of the provided API keys for your solution or register for new ones.

Service	**API Key**
AbuseIPDB	f94b904c9f414170343d0a5a61948dfbc0b55a6eaa0a50a3489e1e66b17ead6527975834e0d
IPQualityScore	mDdEouFdaAG4XvsfCmR6LAWTvARwyRb9

Testing Requirements

We expect you to include basic tests for your application:

- **Frontend:** Unit test for your state logic or component rendering.
- **Backend:** Unit test for your core logic (e.g., IP validation, API call aggregation).
- **Tools:** You can use popular testing frameworks such as Jest, React Testing Library, etc.

Bonus

1. Risk Scoring & Highlighting:
    - Combine fields like abuseConfidenceScore, fraud_score, vpn/proxy flags into a simple “Overall Risk Level”: Low / Medium / High.
    - Show it visually with icons or color coding.
2. Persistent Search History (in-memory or localStorage):
    - Save the user’s last 5–10 lookups locally.
    - Allows users to revisit or compare prior results.
3. Rate Limit Handling (Backend):
    - If the external APIs return 429 or other rate-limit errors, handle them gracefully.
    - Return a clear message to the frontend like: “Rate limit reached, try again later.”

What to Submit

Please provide us with either a **GitHub link** to your repository or a **ZIP archive** containing:

- Full source code for both the frontend and backend.
- A comprehensive `README.md` file that includes:
    - Clear setup instructions for running your application.
    - Guidance on API key configuration.
    - A `.env` file.
- All your test files.
- **Optional:** Any notes about your development approach or tradeoffs you made.

Good luck!