Below are five coding test ideas that require building a **live web app** which interacts with a public API. Each can be done within about an hour (give or take), using **JavaScript/React/Python/PHP** (or any combination). These tests focus on core functionality—API integration, basic UI, and async programming—rather than heavy styling.

Note: Live apps deployed on any platform like 'https://vercel.com/' or 'https://streamlit.io/' or any other of your choice would be given highest weightage, instead of plain github repo links

---

1. Movie Search App using the OMDb API

Requirements:

1. **Search bar**: Users can type a movie title.
2. **Fetch** results from the OMDb API based on the search term.
3. **Display** a list/grid of movie results (poster, title, year).
4. **Detail view**: Clicking on a movie shows more details (plot, actors, etc.).

Key Skills Tested:

- Consuming a third-party REST API with GET requests.
- Managing component or view state for search terms/results.
- Parsing JSON data and rendering it in the UI.

---

2. Weather Dashboard using the OpenWeatherMap API

Requirements:

1. **Search bar**: User enters a city name.
2. **Fetch** the current weather data for that city.
3. **Display** temperature, description, humidity, and so on.
4. Add a **5-day forecast** view.

Key Skills Tested:

- Using an API that requires a key (minimal sign-up required).
- Handling asynchronous requests and loading/error states.
- Parsing JSON to show multiple pieces of data (current weather, forecast).

---

3. Pokémon Info App using the PokéAPI

Requirements:

1. **Search** or **dropdown**: Let users pick a Pokémon by name or ID.
2. **Fetch** data from the PokéAPI.
3. **Display** Pokémon image, name, types, and stats.
4. Navigate to **previous/next** Pokémon.

Key Skills Tested:

- Interacting with a free, no-auth required REST API.
- Rendering images, text, stats in a user-friendly layout.
- Possibly using routing or query parameters in React or a simple server framework.

---

4. Random Cat Images Gallery using The Cat API

Requirements:

1. **Fetch** multiple cat images from The Cat API.
2. **Display** them in a grid or list view.
3. Add a **"Load More"** or **"Refresh"** button to fetch new images.
4. Let user filter by breed.

Key Skills Tested:

- Integrating a public API that returns images.
- Basic layout (gallery, cards).
- Handling parameters (breed filter, limit, etc.).

General Guidelines

1. **Tech Stack**:
    - **Front End**: React or plain JavaScript (or a minimal Flask/PHP template).
    - **Back End** (optional): If you need to proxy requests, use Node/Express, Flask, or simple PHP.

2. **Getting Started**:
    - Use quick-start boilerplates (e.g., Create React App, a minimal Flask/Node template) to save setup time.
3. **Focus**:
    - Make **API calls** to fetch data.
    - **Render** results in a simple, usable interface.
    - Handle **loading** and **errors** gracefully.
    - Optionally add a **search**, **filter**, or simple detail page.
4. **Styling**:
    - Keep it minimal (just enough so the layout is clear).
5. **Testing**:
    - In a 1-hour test, just open in a browser to ensure data is fetched and displayed properly.

These challenges demonstrate your ability to connect to an external API, parse and render data, and implement basic UI functionality in a short timeframe. Good luck!