### **Code Review Report**

#### **1. Code Structure and Readability**

* **Strengths:**
  + Effective use of useState for managing weatherData and city states.
  + API integration is straightforward using axios.
* **Issues:**
  + Typo in setWeaterData (should be setWeatherData), which may confuse collaborators.
  + Hardcoded API key (apikey) exposes sensitive information.
* **Recommendations:**
  + Correct the typo in the state variable setter.
  + Store the API key in a .env file (e.g., const apikey = process.env.REACT\_APP\_WEATHER\_API\_KEY).

#### **2. Functionality**

* **Strengths:**
  + handleChange correctly updates the state for user input.
  + handleData successfully fetches weather data using the OpenWeather API.
* **Issues:**
  + Missing validation for empty city input before API requests.
  + No loading state to inform the user during the API call.
* **Recommendations:**
  + Add input validation to ensure the city name is not empty.
  + Implement a loading spinner or message while waiting for API responses.

#### **3. UX/UI**

* **Strengths:**
  + Simple and functional user interface.
  + Weather data is displayed neatly, including city name, temperature, and weather icon.
* **Issues:**
  + alert for invalid city input is intrusive and outdated.
  + Inline styling for the bar width reduces code readability and reusability.
* **Recommendations:**
  + Replace alert with a user-friendly error message displayed in the UI.
  + Move inline styles to a CSS file or use a CSS-in-JS library for better maintainability.

#### **4. Code Quality and Best Practices**

* **Strengths:**
  + Modular structure with a separate Weather component.
  + Good use of modern React features like hooks.
* **Issues:**
  + Redundant inline temperature conversion logic (weatherData.main.temp - 273).
* **Recommendations:**
  + Extract temperature conversion into a reusable utility function (e.g., convertKelvinToCelsius).
  + Ensure consistent naming conventions (e.g., camelCase for all variables).

#### **5. Performance**

* **Issues:**
  + No debounce mechanism in handleChange, leading to potential API spam.
  + Lack of proper error boundaries in case of incomplete or failed API responses.
* **Recommendations:**
  + Implement debouncing in handleChange to reduce excessive API calls.
  + Add error boundaries to gracefully handle API failures.

#### **6. Suggested Improvements**

* **Security:**
  + Hide the API key using environment variables.
* **User Experience:**
  + Disable the search button while the API call is in progress.
  + Display fallback UI when no data is available or the API call fails.
* **Styling:**
  + Move all inline styles to the CSS file for separation of concerns.
  + Enhance the visual appeal by applying a polished UI design.

**Summary:** The code is functional and provides the basic features expected from a weather application. However, improvements in security (API key management), UX (error handling, loading states), and code quality (typo corrections, reusable functions) will make it more robust and maintainable.