# **Frontend Developer Technical Assessment: Interactive Data Dashboard Widget**

## **Overview**

Create a specific interactive dashboard widget that displays and analyzes weather data across multiple cities. This widget must function as a standalone React application demonstrating your proficiency with component architecture, state management, and data visualization.

## **Detailed Requirements**

### **1. Data Source & Integration**

* Use the OpenWeatherMap API (free tier) with this endpoint format: https://api.openweathermap.org/data/2.5/forecast?q={CITY\_NAME}&appid={API\_KEY}
* Create a mock API service that simulates API calls (to avoid rate limiting during development)
* Handle at least 5 specific cities: London, New York, Tokyo, Sydney, and Cairo
* Implement proper error handling for API failures with specific error messages

### **2. Widget UI Components**

Create these exact components:

* WeatherWidget: Main container component
* CitySelector: Dropdown menu for city selection with search functionality
* WeatherDisplay: Shows current weather data with specific icons
* ForecastList: 5-day forecast displayed as cards
* DataVisualization: Temperature chart using SVG (no chart libraries)
* SettingsPanel: Configurable options panel (units, refresh rate, display options)
* ErrorBoundary: Component that catches and displays errors

### **3. State Management Implementation**

* Create a custom hook called useWeatherData that handles all API calls and data transformation
* Implement a reducer pattern with these exact actions: FETCH\_WEATHER, CHANGE\_CITY, TOGGLE\_UNIT, SET\_ERROR, CLEAR\_ERROR
* Create a Theme context with dark/light mode toggle functionality
* Implement form validation for city search with specific error messages

### **4. Specific JavaScript Challenges**

* Create a data transformation function that:
  + Converts temperature between Celsius and Fahrenheit
  + Calculates daily averages from hourly forecasts
  + Finds temperature min/max/average for the forecast period
  + Implements custom sorting of weather data by date and time
* Create a throttling function for API calls (max 1 call per 5 seconds)
* Implement a custom debounce function for the search input (300ms delay)

### **5. UI Requirements**

* Widget must be exactly 800px wide on desktop and fully responsive on mobile
* Implement a tab system to switch between "Current Weather", "Forecast", and "Statistics" views
* Create custom CSS transitions when switching cities (fade out/in)
* Implement dark/light mode with a specific color palette:
  + Light: #f8f9fa (background), #212529 (text), #0d6efd (accent)
  + Dark: #212529 (background), #f8f9fa (text), #0d6efd (accent)

### **6. Testing Requirements**

* Write unit tests for the useWeatherData hook
* Create test cases for the temperature conversion functions
* Test the debounce functionality
* Implement snapshot tests for the WeatherDisplay component

## **Specific Deliverables**

1. **Component Architecture Diagram**: A visual representation of your component hierarchy and data flow
2. **React Application**: Complete source code meeting all requirements
3. **Documentation**:
   * Setup instructions including how to configure the API key
   * Component documentation with props and state descriptions
   * Explanation of custom hooks and their usage
   * Performance optimization strategies implemented

## 

## **Evaluation Metrics**

1. **Component Structure** (25%)
   * Proper component composition and reusability
   * Effective use of props and state
   * Implementation of required components
2. **State Management** (25%)
   * Correct implementation of the reducer pattern
   * Proper data flow and state updates
   * Effective context usage
3. **JavaScript Proficiency** (25%)
   * Data transformation quality
   * Implementation of utility functions (debounce, throttle)
   * Code organization and modularity
4. **UI Implementation** (15%)
   * Responsive design adherence
   * Theme implementation
   * Animation and transition quality
5. **Testing & Documentation** (10%)
   * Test coverage and quality
   * Documentation completeness and clarity

## **Submission Process**

1. Create a GitHub repository with your solution
2. Include a README.md with all required documentation
3. Ensure the application can run with minimal setup (provide mock data if needed)
4. Submit within 0-4 days of receiving this assessment

## 

**Additional Instructions for Candidates**

1. **Start Date & Deadline**
   * As soon as you receive this test, please reply with the exact date and time you begin working on it.
   * You will have a total of **0–4 days** (96 hours) to complete and submit your solution.
2. **Ease of Launch**
   * Your application must be straightforward to run. Provide clear, step-by-step launch instructions (e.g., “clone the repo, run npm install, set REACT\_APP\_OWM\_API\_KEY, then npm start”).
   * If any dependencies or environment variables are required, document them explicitly so we can start the app without troubleshooting.
3. **Metrics & Quality**
   * We will evaluate your solution based on logical structure, code organization, and how you measure performance (e.g., loading times, render speed, API-call throttling).
   * Wherever possible, include comments or a brief section in your README explaining how you measured and optimized key metrics (for example, how you verified that data-fetching never exceeds one call per 5 seconds).
4. **Submission & Review**
   * Submit your code repository link by the end of your allotted timeframe.
   * After we review your submission, we will reach out to schedule a Zoom meeting.
5. **Zoom Presentation**
   * During the call, be prepared to walk us through:  
     + Your overall **application logic** (why you structured components the way you did).
     + **Data flow** and state management decisions (how useWeatherData and your reducer communicate).
     + Any **trade-offs** or creative solutions you implemented (for example, why you chose a custom SVG chart instead of a library).
     + How you addressed **performance** and **responsiveness** (throttling, debounce, CSS transitions, dark/light theming).
6. **Questions & Support**
   * If you have any questions about the task, send a message via Telegram to **+998 99 971 83 83**. Please include a brief subject line (e.g., “Weather Widget Question”) and describe your issue clearly.
   * We encourage you to ask clarifying questions early rather than guessing requirements.

Good luck! We look forward to seeing your creativity and technical skills.