# Internship Task Report: React.js Developer

**SUBMITTED BY:**

- **Intern Name:** Gopinath U
- **Intern ID:** BS/REG/115245
- **Date of Submission:** January 5, 2026
- **Task Number:** 02
- **Task Title:** React Weather API Project (Precision Forecast)

---

**1. Project Overview**

- **Project Title:** Precision Forecast
- **Domain:** React.js Development
- **Objective:** To develop a real-time weather application that fetches and displays current atmospheric data from the OpenWeatherMap API based on user input.

**2. Technologies Used**

- **Frontend Library:** React.js (Functional Components)
- **State Management:** React useState Hook
- **Data Fetching:** Asynchronous Fetch API (async/await)
- **Styling:** Normal Classic CSS
- **API Provider:** OpenWeatherMap API
- **Deployment:** Netlify

**3. Key Features & Functionality**

- **Dynamic Search:** Allows users to query weather for any city globally using an interactive search bar.
- **Real-Time Metrics:** Displays temperature (formatted to 1 decimal place), humidity, and weather descriptions.
- **Weather Condition Mapping:** Includes a dedicated logic (getWeatherEmoji) that maps API-specific weather IDs to relevant emojis (e.g., ⛈️ for Thunderstorms, ❄️ for Snow).

- **Robust Error Handling:** Implements validation to prevent empty searches and provides "City not found" alerts for invalid queries.
- **Loading Indicators:** Displays a "Loading..." state to improve user experience during data fetching.

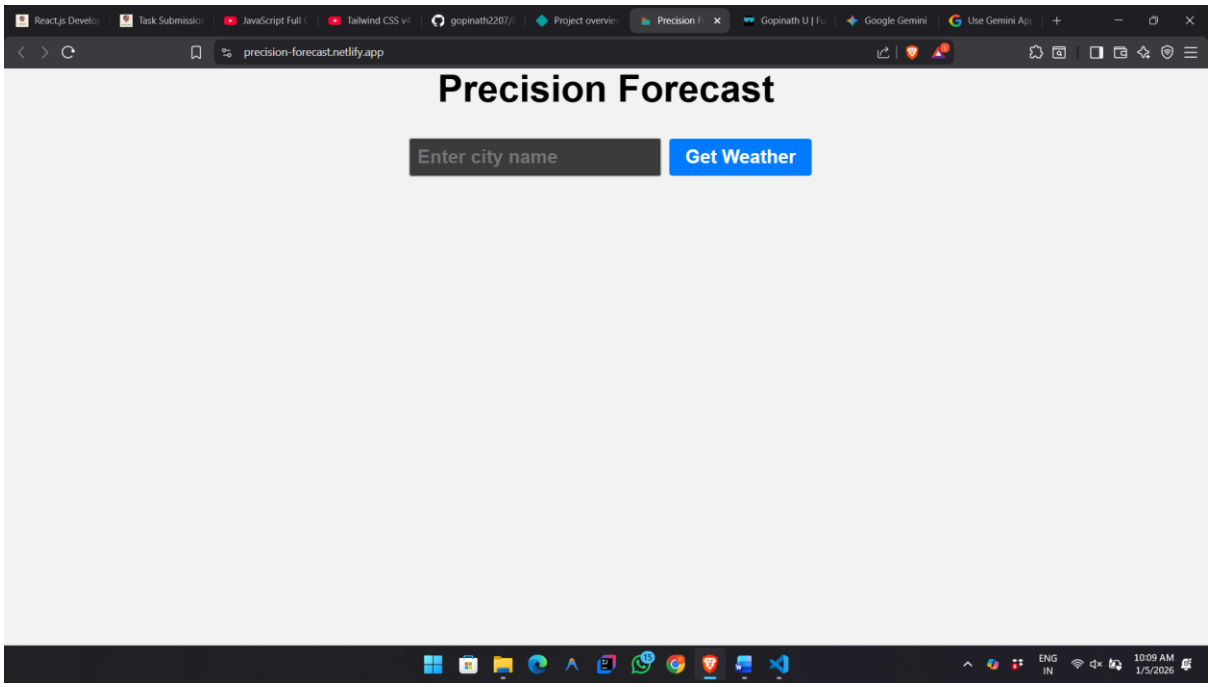
### 4. Technical Implementation Detail

- **Stateful Architecture:** Utilized four separate useState hooks to manage the search input (city), retrieved data (weather), error messages (error), and fetching status (loading).
- **Asynchronous Logic:** Implemented getWeatherData as an async function to handle external API requests without blocking the UI thread.
- **Conditional Rendering:** Used logical && and ternary operators to show/hide the results container based on whether data or errors exist.

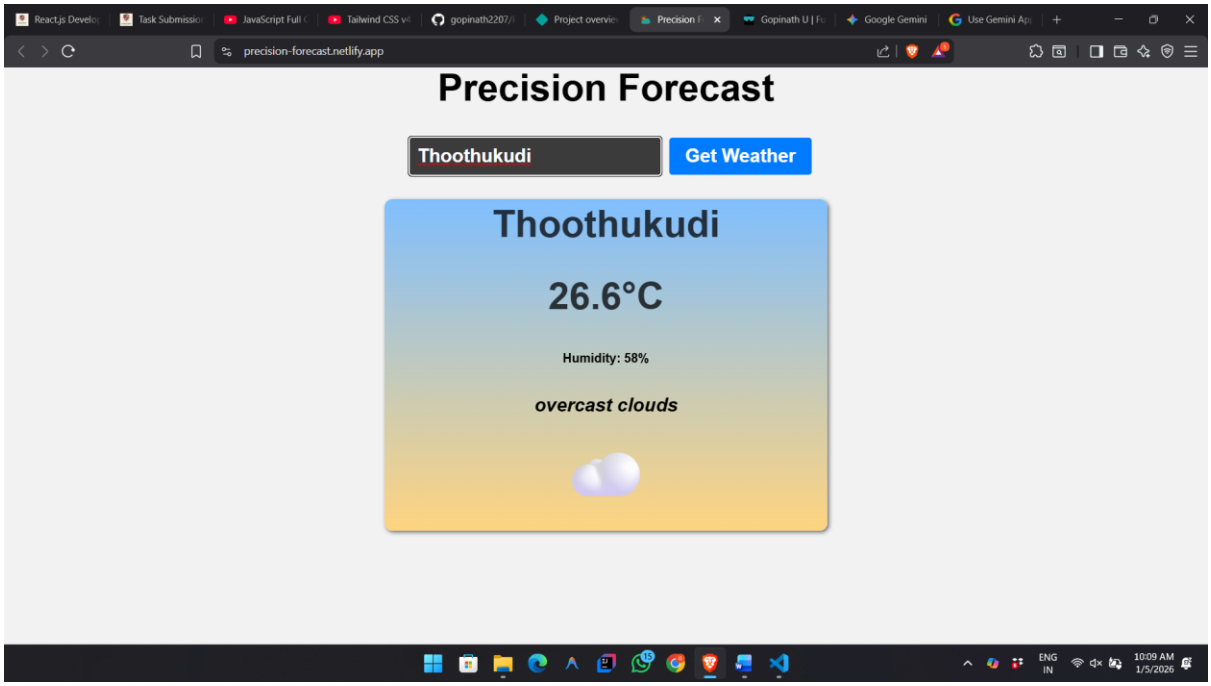
### 5. Core Code Snippet (API Integration)

```
import { useState } from 'react';

function Weather(){
  const [city, setCity] = useState('');
  const [weather, setWeather] = useState(null);
  const [error, setError] = useState('');
  const [loading, setLoading] = useState(false);

  const apiKey = "2e941a95268ab0fdeddcf1030181d984";

  async function getWeatherData(query){
    const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=${encodeURIComponent(query)}&appid=${apiKey}&units=metric`;
    const response = await fetch(apiUrl);
    if(!response.ok){
      throw new Error('City not found');
    }
    return await response.json();
  }

  function getWeatherEmoji(id){
    if(id >= 200 && id < 300){
      return "⛈️"; // Thunderstorm
    }
```

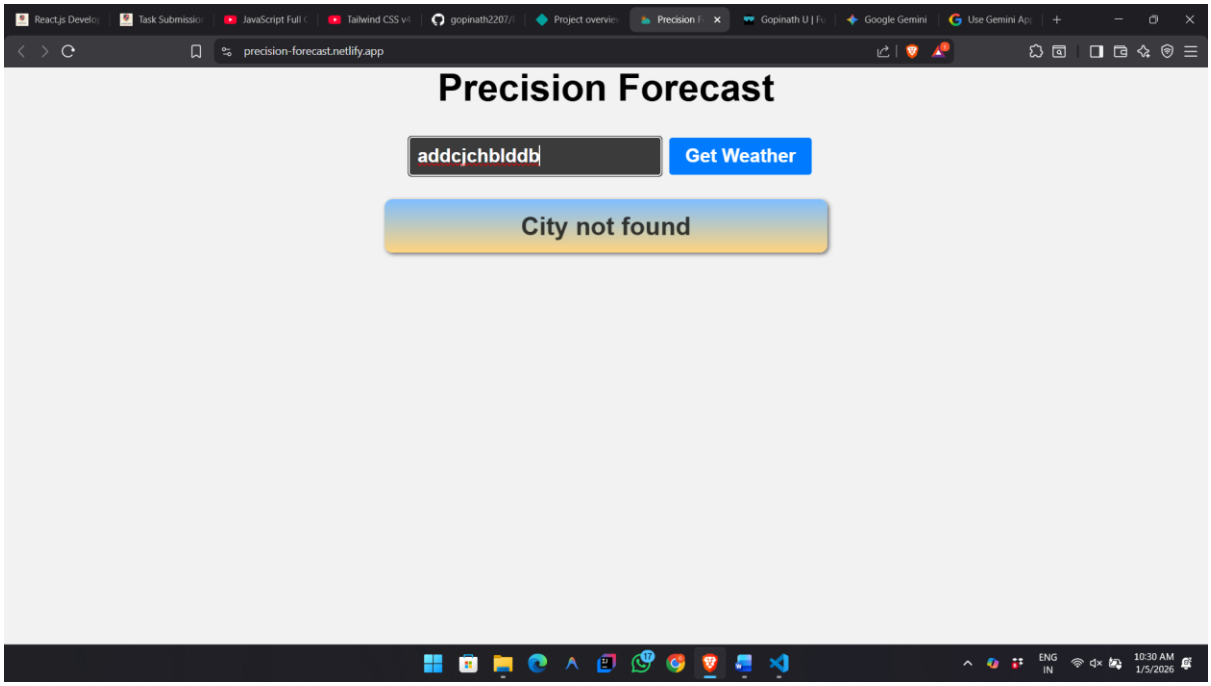
## 6. Screenshots




1. **Initial Interface:** Showing the "Precision Forecast" header and search bar.




2. **Weather Display:** Results showing the temperature, humidity, and condition emoji.




3. **Validation View:** Displaying the "City not found" error message.

**7. Project Links**

- **GitHub Repository:** https://github.com/gopinath2207/WeatherApi
- **Live Hosted Link:** https://precision-forecast.netlify.app/

**8. Conclusion**

This project successfully demonstrates the ability to consume third-party REST APIs within a React environment. By utilizing functional components and hooks, I created a modular and responsive application that provides real-time value to users.

---

**Declaration:**

I, Gopinath, confirm that this project was developed by me as part of the React.js Developer Internship at Alfido Tech.