```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>IBM - FE LIVE WEATHER DASHBOARD</title>
  <style>
    /* Minimal, responsive styling */
    :root{font-family:Inter,system-ui,Segoe UI,Roboto,Arial; --card-bg:#ffffff; --muted:#666;}
    body{margin:0;padding:18px;background:#f3f6fb;color:#0b1220}
    .app{max-width:980px;margin:0 auto;display:grid;grid-template-columns:1fr;gap:14px}
    header{display:flex;justify-content:space-between;align-items:center}
    .card{background:var(--card-bg);padding:14px;border-radius:12px;box-shadow:0 6px 18px
rgba(12,18,30,0.06)}
    .controls{display:flex;gap:8px;flex-wrap:wrap}
    input[type="search"]{padding:8px 12px;border-radius:8px;border:1px solid #ddd;min-
width:220px}
    button{padding:8px 12px;border-
radius:8px;border:none;background:#1f6feb;color:#fff;cursor:pointer}
    button.secondary{background:#6c78a9}
    .row{display:flex;gap:12px;flex-wrap:wrap}
    .weather-main{display:flex;align-items:center;gap:16px}
    .temp{font-size:48px;font-weight:700}
    .meta{color:var(--muted)}
    .forecast{display:flex;gap:10px;flex-wrap:wrap;margin-top:12px}
    .fbox{min-width:110px;padding:10px;border-radius:10px;background:#f7f9ff;text-align:center}
    footer{font-size:13px;color:var(--muted);text-align:center;margin-top:8px}
    @media(min-width:820px){ .app{grid-template-columns:1fr} }
  </style>
</head>
<body>
  <div class="app">
```

```
<header>
  <h2>IBM - FE LIVE WEATHER DASHBOARD</h2>
  <div class="meta">Live • Demo</div>
</header>

<section class="card">
  <div class="controls">
    <input id="searchBox" type="search" placeholder="Enter city name (e.g., Chennai) or 'lat,lon'"/>
    <button id="searchBtn">Search</button>
    <button id="gpsBtn" class="secondary">Use My Location</button>
    <select id="provider">
      <option value="openweather">OpenWeatherMap (demo)</option>
      <option value="ibm">IBM / Weather Company (template)</option>
    </select>
  </div>
</section>

<section id="current" class="card" aria-live="polite">
  <div id="curContent">
    <div class="weather-main">
      <div>
        <div id="place" style="font-weight:600">—</div>
        <div id="desc" class="meta">Search or use location</div>
      </div>
      <div style="margin-left:auto;text-align:right">
        <div id="temp" class="temp">--°C</div>
        <div id="feels" class="meta">Feels like: --</div>
      </div>
    </div>

    <div class="row" style="margin-top:10px">
```

```
          <div>Humidity: <span id="humidity">--%</span></div>
          <div>Wind: <span id="wind">-- m/s</span></div>
          <div>Pressure: <span id="pressure">-- hPa</span></div>
        </div>
        <div id="extra" style="margin-top:10px" class="meta"></div>
      </div>
    </section>

    <section id="forecast" class="card">
      <h4 style="margin:0 0 8px 0">Short Forecast</h4>
      <div id="forecastBoxes" class="forecast"></div>
    </section>

    <section class="card">
      <h4 style="margin:0 0 8px 0">Notes</h4>
      <ol style="margin:0;padding-left:18px" class="meta">
        <li>Provider: <span id="activeProvider">OpenWeatherMap</span></li>
        <li>For production: route requests through a server proxy to hide your API key and avoid CORS
issues.</li>
      </ol>
    </section>

    <footer class="meta">Tip: choose provider, then search city or click "Use My Location".</footer>
  </div>

  <script>
    /**********************************************
     * CONFIG
     * Replace OPENWEATHER_API_KEY with your key.
     * For IBM: this demo includes a template call (commented) using:
```

```
   *
https://api.weather.com/v3/wx/observations/current?geocode={lat},{lon}&units=m&language=en-
US&format=json&apiKey=YOUR_API_KEY
   * See IBM docs for details & fields. (Do NOT expose production keys in client).
   *********************************************/
  const OPENWEATHER_API_KEY = "OPENWEATHER_API_KEY"; // <-- put your OpenWeatherMap
key here
  const elements = {
    place: document.getElementById("place"),
    desc: document.getElementById("desc"),
    temp: document.getElementById("temp"),
    feels: document.getElementById("feels"),
    humidity: document.getElementById("humidity"),
    wind: document.getElementById("wind"),
    pressure: document.getElementById("pressure"),
    forecastBoxes: document.getElementById("forecastBoxes"),
    activeProvider: document.getElementById("activeProvider"),
    extra: document.getElementById("extra")
  };

  // utility: friendly error
  function setError(msg){
    elements.place.textContent = "Error";
    elements.desc.textContent = msg;
    elements.temp.textContent = "--°C";
    elements.forecastBoxes.innerHTML = "";
  }

  // parse lat,lon or city string
  function parseSearch(input){
    input = input.trim();
    const latlon = input.split(",").map(s=>s.trim());
   if(latlon.length===2 && !isNaN(parseFloat(latlon[0])) && !isNaN(parseFloat(latlon[1]))){
    return {lat: parseFloat(latlon[0]), lon: parseFloat(latlon[1])};
   }
   return {q: input};
  }

  // call OpenWeatherMap current + 3-day forecast (using One Call where available)
  async function fetchOpenWeather(lat, lon, q){
   try{
    if(q){
     // search by city name -> get coords
     const url =
`https://api.openweathermap.org/data/2.5/weather?q=${encodeURIComponent(q)}&appid=${OPEN
WEATHER_API_KEY}&units=metric`;
     const res = await fetch(url);
     if(!res.ok) throw new Error('City not found');
     const data = await res.json();
     lat = data.coord.lat; lon = data.coord.lon;
     // reuse data for current display:
     updateCurrentFromOpen(data);
     // then fetch one-call for forecast
    } else {
     // fetch current by coords
     const urlc =
`https://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&appid=${OPENWEATHER
_API_KEY}&units=metric`;
     const resC = await fetch(urlc);
     if(!resC.ok) throw new Error('Unable to fetch current weather');
     const dataC = await resC.json();
     updateCurrentFromOpen(dataC);
    }
```

```
        // OneCall (3 day) - note OneCall 3.0 may be on different path; this is a general approach
        const oneCall =
`https://api.openweathermap.org/data/2.5/onecall?lat=${lat}&lon=${lon}&exclude=minutely,hourly,
alerts&appid=${OPENWEATHER_API_KEY}&units=metric`;
        const r = await fetch(oneCall);
        if(!r.ok) throw new Error('Unable to fetch forecast');
        const forecast = await r.json();
        updateForecastFromOpen(forecast);
      }catch(err){
        console.error(err);
        setError(err.message || 'OpenWeather error');
      }
    }

    function updateCurrentFromOpen(data){
      const place = `${data.name || data?.sys?.country || ''}`.trim();
      elements.place.textContent = place || `${data.coord.lat.toFixed(2)},${data.coord.lon.toFixed(2)}`;
      const weather = data.weather && data.weather[0];
      elements.desc.textContent = weather ? `${weather.main} — ${weather.description}` : '—';
      elements.temp.textContent = `${Math.round(data.main.temp)}°C`;
      elements.feels.textContent = `Feels like ${Math.round(data.main.feels_like)}°C`;
      elements.humidity.textContent = `${data.main.humidity}%`;
      elements.wind.textContent = `${(data.wind.speed||0)} m/s`;
      elements.pressure.textContent = `${data.main.pressure} hPa`;
      elements.extra.textContent = `Updated: ${new
Date((data.dt||Date.now())*1000).toLocaleString()}`;
    }

    function updateForecastFromOpen(fore){
      // show next 3 days (today + 2). daily[0] = today
      const days = (fore.daily || []).slice(0,4);
      elements.forecastBoxes.innerHTML = days.map(d=>{
```

```
      const dt = new Date(d.dt * 1000);
      const day = dt.toLocaleDateString(undefined,{weekday:'short',month:'short',day:'numeric'});
      const icon = d.weather && d.weather[0] && d.weather[0].icon;
      const iconUrl = icon ? `https://openweathermap.org/img/wn/${icon}@2x.png` : '';
      return `
        <div class="fbox">
          <div style="font-size:13px;font-weight:600">${day}</div>
          <div style="margin:6px 0"><img src="${iconUrl}" alt="" width="60" height="60"></div>
          <div style="font-weight:700">${Math.round(d.temp.day)}°C</div>
          <div class="meta">Min ${Math.round(d.temp.min)}° • Max
${Math.round(d.temp.max)}°</div>
        </div>
      `;
    }).join("");
  }

  /********* IBM Weather Company example (template) *********
  * Example endpoint (current conditions by geocode):
  * https://api.weather.com/v3/wx/observations/current?geocode=40.58,-
111.66&units=m&language=en-US&format=json&apiKey=YOUR_API_KEY
  *
  * If you choose provider = "ibm", you must:
  *  - Obtain an IBM/Weather Company API key.
  *  - Ensure your key has permission for that atomic endpoint.
  *  - Proxy the request server-side (recommended) to keep the key secret and avoid CORS.
  *
  * Example fetch (client-side demo only — not for production):
  *
  async function fetchIBM(lat, lon){
    const apiKey = "YOUR_IBM_KEY"; // don't expose in production
```

```
      const url =
`https://api.weather.com/v3/wx/observations/current?geocode=${lat},${lon}&units=m&language=en
-US&format=json&apiKey=${apiKey}`;

      const res = await fetch(url);

      if(!res.ok) throw new Error("IBM Weather request failed");

      const payload = await res.json();

      // payload fields differ from OpenWeather; see IBM docs for mapping

      // Example mapping:

      // payload.temperature, payload.wxPhraseLong, payload.relativeHumidity

    }

    *******************************************************/


    // handle UI actions

    document.getElementById("searchBtn").addEventListener("click", onSearch);

    document.getElementById("searchBox").addEventListener("keyup", (e)=>{ if(e.key==='Enter')
onSearch(); });

    document.getElementById("gpsBtn").addEventListener("click", useGeo);

    document.getElementById("provider").addEventListener("change", (e)=>{

      const p = e.target.value;

      elements.activeProvider.textContent = p === 'openweather' ? 'OpenWeatherMap (demo)' : 'IBM /
Weather Company (template)';

    });


    async function onSearch(){

      const provider = document.getElementById("provider").value;

      const input = document.getElementById("searchBox").value.trim();

      if(!input){ setError('Enter city or coordinates'); return; }

      const parsed = parseSearch(input);

      if(provider === 'openweather') {

        if(parsed.q) await fetchOpenWeather(null,null, parsed.q);

        else await fetchOpenWeather(parsed.lat, parsed.lon, null);

      } else {
        // provider = ibm: we show template behavior using IBM endpoint
        // For demo, we'll still try OpenWeather if no IBM key available.
        setError('IBM provider selected: use server proxy with IBM key (see code comments). For quick
demo, switch provider to OpenWeatherMap.');
      }
    }

    function useGeo(){
      if(!navigator.geolocation){ setError('Geolocation not supported'); return; }
      navigator.geolocation.getCurrentPosition(async (pos)=>{
        const lat = pos.coords.latitude, lon = pos.coords.longitude;
        const provider = document.getElementById("provider").value;
        if(provider === 'openweather') await fetchOpenWeather(lat, lon, null);
        else setError('IBM provider selected: use a server proxy for IBM calls (see docs).');
      }, (err)=>{
        setError('Location permission denied or unavailable');
      }, {timeout:8000});
    }

    // small demo start: show default city
    (async ()=>{ document.getElementById("searchBox").value = "Chennai"; await onSearch(); })();

  </script>
</body>
</html>
```