```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Elegant Weather UI</title>
  <!-- Tailwind CSS CDN for styling -->
  <script src="https://cdn.tailwindcss.com"></script>
  <!-- Google Fonts - Inter for a clean look -->
  <link
href="https://fonts.googleapis.com/css2?family=Inter:wght@400;500;600;700&display=swap"
rel="stylesheet">
  <style>
    /* Custom styles for the body and to hide scrollbar */
    body {
      font-family: 'Inter', sans-serif;
      background: linear-gradient(to bottom right, #0f172a, #1e293b); /* Dark gradient background
*/
      color: white; /* Default text color */
      min-height: 100vh; /* Ensure body takes full viewport height */
      display: flex;
      align-items: center;
      justify-content: center;
      /* No padding on body; padding is on the main container for better control */
    }
    /* Remove default borders from input and button for a cleaner look */
    input[type="text"] {
      border: none;
    }
    button {
      border: none;
    }
    /* Custom CSS to hide the scrollbar while keeping its functionality */
    .hide-scrollbar::-webkit-scrollbar {
        display: none; /* For Chrome, Safari, and Opera */
    }
    .hide-scrollbar {
        -ms-overflow-style: none;  /* For Internet Explorer and Edge */
        scrollbar-width: none;  /* For Firefox */
    }
  </style>
</head>
<body class="flex items-center justify-center min-h-screen">
  <!-- Main weather card container -->
```

```
  <!-- Adjusted max-w and min-w for an even smaller overall size -->
  <div class="bg-[#0f172a] rounded-xl shadow-xl max-w-lg w-full p-3.5 md:p-4 lg:p-5
min-w-[280px]">
    <!-- Top Section: City Name, Date/Time, Current Temperature, Description -->
    <div class="flex flex-col md:flex-row md:justify-between md:items-start">
      <div class="flex-shrink-0">
        <h1 id="cityName" class="text-base font-semibold">City Name</h1> <!-- Reduced font
size -->
        <p id="dateTime" class="text-xs text-gray-400">-- Date Time --</p>
        <div class="flex items-center mt-1"> <!-- Adjusted margin-top -->
          <div class="text-3xl font-bold" id="temperature">--°C</div> <!-- Reduced font size -->
          <div class="ml-1 text-gray-400 text-xs">°C | °F</div> <!-- Adjusted font size -->
        </div>
        <p id="description" class="capitalize text-xs">--</p> <!-- Reduced font size -->
      </div>

      <!-- Search Bar, Settings Icon, Current Weather Details (Feels like, Humidity, Wind) -->
      <div class="flex flex-col items-end space-y-1 mt-2 md:mt-0 md:ml-2"> <!-- Adjusted spacing
-->
        <div class="flex items-center space-x-1"> <!-- Adjusted spacing -->
          <input id="weather-city" placeholder="Enter city name" class="px-1.5 py-0.5 rounded-md
bg-gray-700 placeholder-gray-400 focus:outline-none w-32 text-xs" /> <!-- Adjusted padding and
width -->
          <button id="get-weather" class="bg-blue-600 hover:bg-blue-700 px-1.5 py-0.5
rounded-md text-xs">Search</button> <!-- Adjusted padding and font size -->
          <div class="w-5 h-5 bg-gray-700 rounded-md flex items-center justify-center
cursor-pointer"> <!-- Adjusted size -->
            <!-- SVG for settings icon (a gear icon) -->
            <svg class="w-3 h-3 text-gray-400" fill="none" stroke="currentColor" viewBox="0 0 24
24" xmlns="http://www.w3.org/2000/svg"><path stroke-linecap="round" stroke-linejoin="round"
stroke-width="2" d="M10.325 4.317c.426-1.756 2.924-1.756 3.35 0a1.724 1.724 0 002.573
1.066c1.543-.942 3.33.832 2.76 2.375a1.724 1.724 0 001.065 2.572c1.756.426 1.756 2.924 0
3.35a1.724 1.724 0 00-1.066 2.573c.942 1.543-.832 3.33-2.375 2.76a1.724 1.724 0 00-2.572
1.065c-.426 1.756-2.924 1.756-3.35 0a1.724 1.724 0
00-2.573-1.066c-1.543.942-3.33-.832-2.76-2.375a1.724 1.724 0
00-1.065-2.572c-1.756-.426-1.756-2.924 0-3.35a1.724 1.724 0
001.066-2.573c-.942-1.543.832-3.33 2.375-2.76a1.724 1.724 0 002.572-1.065z"></path><path
stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M15 12a3 3 0 11-6 0 3 3 0
016 0z"></path></svg>
          </div>
        </div>
        <div class="flex items-center justify-end space-x-1"> <!-- Adjusted spacing -->
          <img id="weatherIcon" src="https://openweathermap.org/img/wn/01d@2x.png"
class="w-12 h-12" alt="Weather Icon" /> <!-- Reduced size -->
```

```
        <div class="text-xs text-gray-300 space-y-0.5"> <!-- Adjusted font size and spacing -->
          <p id="feelsLike">Feels like: --°C</p>
          <p id="humidity">Humidity: --%</p>
          <p id="wind">Wind: -- km/h</p>
        </div>
      </div>
    </div>
  </div>

  <!-- Divider line -->
  <div class="border-t border-gray-600 my-2.5"></div> <!-- Adjusted margin -->

  <!-- Forecast Section - Configured for horizontal sliding -->
  <div id="forecast" class="flex flex-nowrap overflow-x-auto gap-1 py-0.5 hide-scrollbar"> <!--
Adjusted gap and padding -->
    <!-- Forecast items will be dynamically injected here by JavaScript -->
  </div>
 </div>

 <script>
  // **IMPORTANT**: Replace 'YOUR_API_KEY' with your actual OpenWeatherMap API key.
  // Get your API key from: https://openweathermap.org/api
  const apiKey = '4fd53d72eee8ee4cb0bb86e5bda7165f';

  // Get references to HTML elements
  const cityInput = document.getElementById('weather-city');
  const searchBtn = document.getElementById('get-weather');
  const currentWeather = {
    city: document.getElementById('cityName'),
    temp: document.getElementById('temperature'),
    desc: document.getElementById('description'),
    feelsLike: document.getElementById('feelsLike'),
    humidity: document.getElementById('humidity'),
    wind: document.getElementById('wind'),
    dateTime: document.getElementById('dateTime'),
    icon: document.getElementById('weatherIcon'),
  };
  const forecastDiv = document.getElementById('forecast');

  /**
   * Fetches weather data for a given city and updates the UI.
   * @param {string} city - The name of the city to fetch weather for.
   */
  const fetchWeatherData = async (city) => {
```

```
    try {
      // Fetch current weather data
      const currentRes = await fetch(
`https://api.openweathermap.org/data/2.5/weather?q=${encodeURIComponent(city)}&units=metric&appid=${apiKey}`
      );
      const currentData = await currentRes.json();

      // Check for API errors (e.g., city not found, invalid API key)
      if (currentData.cod !== 200) {
        throw new Error(currentData.message || 'City not found or API error.');
      }

      // Update current weather display
      currentWeather.city.textContent = currentData.name;
      currentWeather.temp.textContent = `${Math.round(currentData.main.temp)}°C`;
      currentWeather.desc.textContent = currentData.weather[0].description;
      currentWeather.feelsLike.textContent = `Feels like: ${Math.round(currentData.main.feels_like)}°C`;
      currentWeather.humidity.textContent = `Humidity: ${currentData.main.humidity}%`;
      currentWeather.wind.textContent = `Wind: ${currentData.wind.speed} km/h`;

      // Format and display current date and time
      const now = new Date();
      const dateOptions = { weekday: 'long', year: 'numeric', month: 'long', day: 'numeric' };
      const timeOptions = { hour: 'numeric', minute: 'numeric', hour12: true };
      currentWeather.dateTime.textContent = `${now.toLocaleDateString('en-US', dateOptions)} | ${now.toLocaleTimeString('en-US', timeOptions)}`;

      // Set current weather icon
      currentWeather.icon.src = `https://openweathermap.org/img/wn/${currentData.weather[0].icon}@2x.png`;

      // Fetch 5-day weather forecast data
      const forecastRes = await fetch(
`https://api.openweathermap.org/data/2.5/forecast?q=${encodeURIComponent(city)}&units=metric&appid=${apiKey}`
      );
      const forecastData = await forecastRes.json();

      // Filter forecast data to get one entry per day, ideally around midday (12:00:00)
      const dailyForecasts = [];
```

```
        const seenDates = new Set(); // To track dates for unique daily entries

        forecastData.list.forEach(item => {
            const date = new Date(item.dt * 1000); // Convert Unix timestamp to Date object
            const dateString = date.toISOString().split('T')[0]; // Get YYYY-MM-DD string

            // Add the forecast item if it's a new day or if it's close to midday for that day
            // This helps ensure we get one representative forecast per day
            if (!seenDates.has(dateString) || (date.getHours() >= 11 && date.getHours() <= 13)) {
                dailyForecasts.push(item);
                seenDates.add(dateString);
            }
        });

        forecastDiv.innerHTML = ''; // Clear previous forecast content

        // Display the first 5 days of the forecast
        dailyForecasts.slice(0, 5).forEach(day => {
          const date = new Date(day.dt_txt); // Use dt_txt for consistency with OpenWeatherMap's
string format
          const weekday = date.toLocaleDateString('en-US', { weekday: 'long' });

          // Append HTML for each forecast day to the forecast container
          forecastDiv.innerHTML += `
            <div class="bg-[#1e293b] p-1.5 rounded-xl flex flex-col items-center flex-shrink-0
w-[75px] text-center"> <!-- Adjusted padding and width -->
              <p class="font-semibold text-white mb-0.5 text-xs">${weekday}</p> <!-- Adjusted font
size -->
              <img src="https://openweathermap.org/img/wn/${day.weather[0].icon}@2x.png"
class="w-8 h-8" alt="Weather Icon" /> <!-- Adjusted size -->
              <p class="text-white text-xs">${Math.round(day.main.temp_max)}° /
${Math.round(day.main.temp_min)}°</p> <!-- Adjusted font size -->
              <p class="text-[0.6rem] text-gray-300 capitalize
mt-0.5">${day.weather[0].description}</p> <!-- Adjusted font size -->
            </div>
          `;
        });
      } catch (err) {
        // Display an alert for the user and log the error to the console
        alert(`Error fetching weather data: ${err.message}. Please check the city name or your API
key.`);
        console.error('Weather fetching error:', err);
      }
    };
```

```
    // Event listener for the search button click
    searchBtn.addEventListener('click', () => {
      const city = cityInput.value.trim(); // Get city name from input and remove whitespace
      if (city) {
        fetchWeatherData(city); // Fetch data if city name is provided
      } else {
        alert('Please enter a city name to search.'); // Prompt user if input is empty
      }
    });

    // Initial data load when the DOM is fully loaded
    document.addEventListener('DOMContentLoaded', () => {
      fetchWeatherData('Toronto'); // Load weather for Toronto by default
    });
  </script>
</body>
</html>
```