



## PROJECT SPECIFICATION

**Weather Journal App****Project Environment Setup**

CRITERIA	MEETS SPECIFICATIONS
Node and Express Environment	<p>Node and Express should be installed on the local machine. The project file <code>server.js</code> should require <code>express()</code>, and should create an instance of their app using express.</p> <p>The Express app instance should be pointed to the project folder with <code>.html</code>, <code>.css</code>, and <code>.js</code> files.</p>
Project Dependencies	<p>The 'cors' package should be installed in the project from the command line, required in the project file <code>server.js</code>, and the instance of the app should be setup to use <code>cors()</code>.</p> <p>The <code>body-parser</code> package should be installed and included in the project.</p>
Local Server	<p>Local server should be running and producing feedback to the Command Line through a working callback function.</p>

CRITERIA	MEETS SPECIFICATIONS
API Credentials	Create API credentials on OpenWeatherMap.com

## APIs and Routes

CRITERIA	MEETS SPECIFICATIONS
APP API Endpoint	There should be a JavaScript Object named <code>projectData</code> initiated in the file <code>server.js</code> to act as the app API endpoint.
Integrating OpenWeatherMap API	<p>The personal API Key for OpenWeatherMap API is saved in a named <code>const</code> variable.</p> <p>The API Key variable is passed as a parameter to <code>fetch()</code> .</p> <p>Data is successfully returned from the external API.</p>
Return Endpoint Data	There should be a GET route setup on the server side with the first argument as a string naming the route, and the second argument a callback function to return the JS object created at the top of server

GET Route I: Server Side CRITERIA	code.  MEETS SPECIFICATIONS
Return Endpoint Data  GET Route II: Client Side	There should be an asynchronous function to fetch the data from the app endpoint
POST Route	<p>You should be able to add an entry to the project endpoint using a POST route setup on the server side and executed on the client side as an asynchronous function.</p> <p>The client side function should take two arguments, the URL to make a POST to, and an object holding the data to POST.</p> <p>The server side function should create a new entry in the apps endpoint (the named JS object) consisting of the data received from the client side POST.</p>

## Dynamic UI

CRITERIA	MEETS SPECIFICATIONS

CRITERIA	MEETS SPECIFICATIONS
Naming HTML Inputs and Buttons For Interaction	<p>The <code>input</code> element with the <code>placeholder</code> property set to "enter zip code here" should have an <code>id</code> of <code>zip</code>.</p> <p>The <code>textarea</code> included in project HTML should have an <code>id</code> of <code>feelings</code>.</p> <p>The button included in project HTML should have an <code>id</code> of <code>generate</code>.</p>
Assigning Element Properties Dynamically	<p>The div with the <code>id</code>, <code>entryHolder</code> should have three child divs with the ids:</p> <ul style="list-style-type: none"><li>• <code>date</code></li><li>• <code>temp</code></li><li>• <code>content</code></li></ul>
Event Listeners	<p>Adds an event listener to an existing HTML button from DOM using Vanilla JS.</p> <p>In the file <code>app.js</code>, the element with the <code>id</code> of <code>generate</code> should have an <code>addEventListener()</code> method called on it, with <code>click</code> as the first parameter, and a named callback function as the second parameter.</p>

CRITERIA	MEETS SPECIFICATIONS
Dynamically Update UI	<p>Sets the properties of existing HTML elements from the DOM using Vanilla JavaScript.</p> <p>Included in the async function to retrieve that app's data on the client side, existing DOM elements should have their <code>innerHTML</code> properties dynamically set according to data returned by the app route.</p>