UDACITY

PROJECT SPECIFICATION

Weather Journal App

Project Environment Setup

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

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 `projectData` initiated in the file `server.js` to act as the app API endpoint.
Integrating OpenWeatherMap API	The personal API Key for OpenWeatherMap API is saved in a named `const` variable. The API Key variable is passed as a parameter to `fetch()` . Data is successfully returned from the external API.
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

CRITERIA	MEETS SPECIFICATIONS
GET Route I: Server Side	code.
Return Endpoint Data GET Route II: Client Side	There should be an asynchronous function to fetch the data from the app endpoint
POST Route	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. The client side function should take two arguments, the URL to make a POST to, and an object holding the data to POST. 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.

Dynamic UI

CRITERIA	MEETS SPECIFICATIONS

CRITERIA	MEETS SPECIFICATIONS
Naming HTML Inputs and Buttons For Interaction	The `input` element with the `placeholder` property set to “enter zip code here” should have an `id` of `zip`. The `textarea` included in project HTML should have an `id` of `feelings`. The button included in project HTML should have an `id` of `generate`.
Assigning Element Properties Dynamically	The div with the `id`, `entryHolder` should have three child divs with the ids: • `date` • `temp` • `content`
Event Listeners	Adds an event listener to an existing HTML button from DOM using Vanilla JS. In the file `app.js`, the element with the `id` of `generate` should have an `addEventListener()` method called on it, with `click` as the first parameter, and a named callback function as the second parameter.

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