1. Approach and Thought Process

The overall architecture of the application was first analyzed. Then, the flow of data from user’s input in the Search component up to the App component was traced. From the API call, the response data was observed on how to display its results. Then ‘axios’ was used for API calls as it’s a good choice for promise-based structure. The error handling was done with a try/catch block to either display an error message to the user or in console throughout the program during the development phase.

The core parts of the project were then separated.

* State management: A parent component App.jsx was used to hold a global state so it can be shared between its siblings. If state was stored in the Search component, it would not be possible to pass the ‘WeatherBackground’ component easily.
* User input and data fetch: A component to handle the search operation, make API calls and display the result statistics of weather.
* UI render: Change in state for updating the visuals.

1. Decisions Made

* App.jsx was made the top-level component that handles the application state.
* Implementation of toggle between Celsius and Fahrenheit made a good user experience.
* Since the API returns temperature in Celsius, ‘convertTemp’ function was created to convert values for display.
* API returns time in UNIX timestamps. So, isDay’ was calculated to check if the current time(dt) was between sunrise and sunset to set a Boolean value for either day or night.
* Lookup objects(gifs) were created to map the condition code.
* A ‘default’ asset was created to ensure that the web app always has a background even if the API returns unexpected results.