**Technical assessment**

Create an NodeJS express app which will call an api to get the weather forecast information for an specific area.

A user or external system will be able to update the desired location by providing longitude and latitude values when making a request to the location endpoint listed below. These values should be stored against a user/system which will be used in subsequent calls to the forecast endpoint.

The request to the forecast endpoint should return

* the summary of the current weather
* the wind speed and direction
* An array of the day’s temperature to track on a graph (timestamp and temperature)

Use Postman or similar tool to view the response format, then use only the data required (preferably typed as listed above).

TIP: The response includes many ‘blocks’ of information and the required values can be found across the ‘currently’ and ‘daily’ blocks. Use the Date constructor or other library to parse the time field of the values you want to use.

The application should have these two endpoints:

* **POST /api/location**
* Accept longitude and latitude to store against a user/system.
* **GET /api/forecast**
* Should make a call to the DarkSky API using the longitude and latitude stored against the user/system.

An example request to DarkSky:

GET - [**https://api.darksky.net/forecast/{key}/{latitude},{longitude}?units=si**](https://api.darksky.net/forecast/%7bkey%7d/%7blatitude%7d,%7blongitude%7d?units=si)

In the example above, the units=si means to report temperature in degrees Celsius and distances in kilometres.

Use Google or Apple maps to find longitude and latitude of your preferred city.

Use this DarkSky key to make requests:

**5bd31fc5ee7b0a2aa3df023338e856eb**

You will not be required to implement a database; implementing functions which would be able to handle CRUD operations is sufficient (in memory storage of values are fine).

Keep the implementation simple with the following in mind:

* Use an organised project structure
* Use Express framework
* Use Axios or similar to make downstream requests
* Use in memory data persistence
* Use TypeScript, define data types etc…

**Bonus points:**

* + Write unit tests with data mocking
  + Caching (again, actual integration with caching server is not important, show understanding of caching strategies - use in memory caching)
  + Store configurable values in environment variables
  + Create a simple Dockerfile
  + Use middleware for cross cutting concerns (error handling and logging)