REACT – Node.JS Challenge

WEATHER FORECAST APPLICATION

Your challenge will be to develop a single page JavaScript application that displays the weather forecast for the current week, in different locations.

Step 1 – Front End User Interface

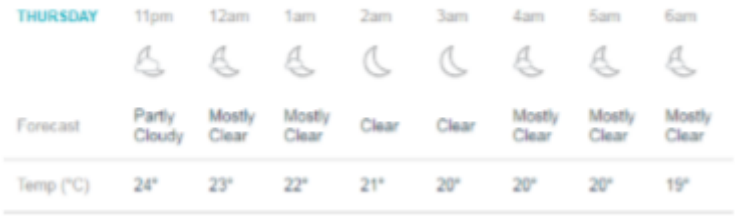
The User Interface will contain at least these elements:

City selector – To search the weather forecast for a city

(a basic one could be a drop down with some prefixed cities)

Week forecast – Diagram displaying the temperature range along the week. As a minimum, it will contain the date, and the temperature range for every day.

Detailed view for a day – In case a specific day is selected, some additional information will be shown (e.g. hourly temperature forecast)



Front End can include more elements. The pictures above are just illustrative (feel free to define a basic UI and then keep improving it along the exercise). As a starting point, some JSON files can be manually created to “mock” the data (please, think about the structure you will use in the next steps).

Step 2 – Node.js Express framework – REST API

In this step, a REST API service must be created to provide weather information using the Nodejs Express framework. You will need to add functionality to add and delete information as well as to perform some queries. You'll be dealing with typical information for weather data like cities, dates, temperature intervals, etc. Please, consider filtering and ordering, response codes and error messages for the queries you must implement.

A very basic JSON structure could look like this one (just illustrative, feel free to define a new one with more information):

```
{
   "id": 1,
   "date": "1985-01-01",
   "location": {
       "city": "Zaragoza",
       "country": "Spain"
   },
   "hourly_temperature": [
      28.5, 27.6, 26.7, 25.9, 25.3, 24.7,
      24.3, 24.0, 27.1, 34.0, 38.6, 41.3,
      43.2, 44.4, 45.0, 45.3, 45.1, 44.2,
      41.9, 38.0, 35.0, 33.0, 31.1, 29.9
   ]
}
```

The REST service should implement the following functionalities:

1. Erasing all the weather data.
2. Adding new weather data.
3. Returning all the weather data.
4. Returning the weather data filtered by date and location.

Data persistence:

A basic option could be to use local storage, but we would prefer that the JSON information is stored in a database. To run the solution locally, we will need some instructions (see RULES section at the end of this document).

EXTRA – bonus mark

Cloud deployment of the whole solution. Technologies/resources to be considered:
- Heroku: NodeJS, static Express server for frontend (https://www.heroku.com)
- Firebase: database (https://firebase.google.com/)
- GitHub Pages: static frontend (https://pages.github.com/)
- Netlify: static frontend (https://www.netlify.com/)

EXTRA – bonus mark

The proposed API is very basic. Some other Open APIs can be taken as a reference to improve it. An API like “Open weather API”, https://openweathermap.org/, can be used.

Example of JSON structure provided by the API:

https://openweathermap.org/data/2.5/forecast/?appid=b6907d289e10d714a6e88b30761fae22&id=31 04324&units=metric

This endpoint can be used in order to find the city id:

https://openweathermap.org/data/2.5/find?q=Zaragoza&cnt=30&appid=b6907d289e10d714a6e88b30 761fae22

EXTRA – bonus mark

Step 3 – “Real time” Front End application

Create a “real time” Front End application that shows latest changes in the weather forecast. Suggestions:

• Node.js could use SSE (Server Sent Events) technology to push updates to the client application

• Any mechanism to implement a periodic call (every 1 min, for example) to the back end (please, feel free to define your own solution).

Rules

1. You should develop the Front End as a Single Page Application using React.
2. Backend integration will be implemented using Node.js Express framework.
3. Create a README.md explaining how to build/run/use the app.
4. Think about control of errors when calling the APIs (404, 500, … response codes)
5. This is an evaluation exercise. The more steps are completed, the better so we can evaluate different skillsets (it is preferred a complete solution rather than just a perfect Front End).
6. If you're done check in your solution into any public git repo hoster (github, bitbucket, etc.) and send us the link

Additional considerations (EXTRA – bonus mark).

7. Think about how you are ensuring quality assurance in your process. 8. Think about browser compatibility (including mobile devices)