Friday, 23 June 2023

I have spent a couple of days going over this task and implementing. I hope it has all the expected functionality. I believe I have covered all that was requested within the specification.

I have used Laravel with Sail, Breeze and Inertia. Inertia allows us to utilise the full power of React within Laravel.

To run….
Pull the repo.
Navigate to the root of the repo.
Run ***composer install***
Run ***npm install***
Then once deps have been installed you should be able to view the front end by running ***rpm run dev*** or ***npm run build***.
Access the application at ***http://localhost***

EUA Weather app

I have taken an iterative approach to this project. There is a huge amount that could be done within the scope of the requirements.

Proposed Project Phases:

- **Phase 1:**
  Simple MVP application to cover specifications given.

- **Phase 2:**
  Integrate Google Maps to allow selection of location by dropping a pin as another option.
  Improve text entry by adding dropdown of pre-filled city options and pass cords directly.
  Implement restrictions on number of uses per user per day if required.
- *Add ability to remove favourites.*

- **Phase 3:**
  Integrate Google Maps to show the weather graphically as well as in text format.

- **Phase 4?:**
  Probably find someone way more imaginative than me to create the front end properly :-)

Considerations

- Choose best API for purpose.
  Not all API's are equal :-)
- *Plugin for Geolocation.*
  Need to find a suitable plugin to use for retrieval of co-ords and place names.
- Plugin for dropdown city search. (Phase 2)
- Back end needs to geolocate user and pass to front end for initial weather load.
- API call via client/js.
- Mocking of API calls/responses for testing.
  This will need to be done for as development and testing could cause potentially hundreds or thousands of extra API calls if not addressed.
- Keeping server load to a minimum.
  This is always a consideration for any application.
- Keeping DB data stored to a minimum.
  There is little data here that we actually need to store at our end. I will only store the user's preferences server side and always pull fresh weather data.
- Minimising the number of API calls.
  In the event a user was overusing the service (constantly refreshing, visiting for the Nth time per day) we could impose a limitation on the particular user. This would I guess be dependent on the API restrictions if any. This would be a phase 2 requirement I would imagine but that would be a stakeholder's decision.

Phase 1 Wireframe.