Instructions:

Time Limit: 48 hours

In this task, you are asked to build a simple Weather WebApp. Static resources are provided in the project folder.

In general, there is no limitation on tools you choose except that **you must use React as front, NodeJS as backbone for the backend (the actual choice of web framework is up to you)**. However, we will evaluate your tools/libraries based on which we give different scores. Preferred tools are pointed out in the technical requirements. Note that completion of the features and meet the technical requirements is the top priority.

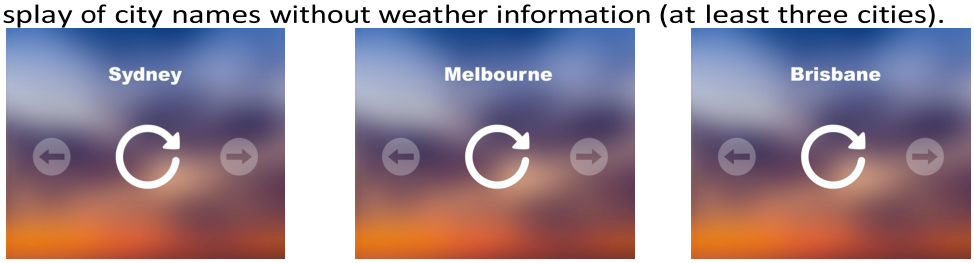
Please carefully read the following requirements before you start.

Requirements:

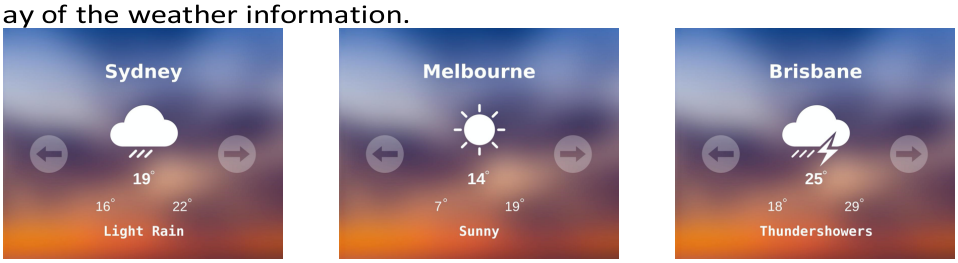
1. *Feature Requirements:*

This Web App contains 3 stages:
First stage contains only a button to start the app. When the button is clicked, the App enters the second stage.

Second stage is a display of city names without weather information (at least three cities).



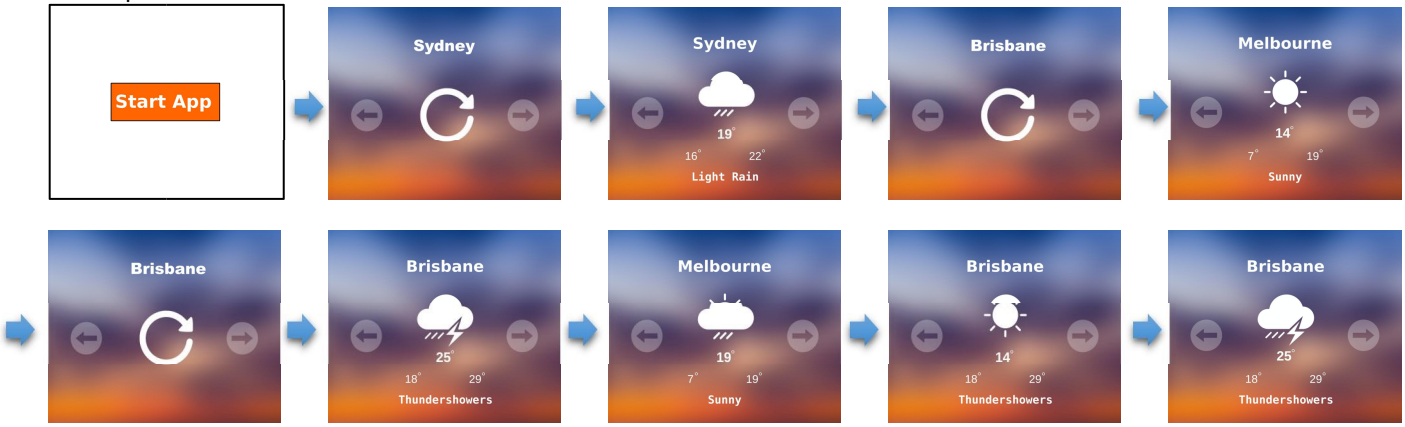
Third stage is a display of the weather information.



During interaction, Stage 2 and Stage 3 are interleaved.

Clicking the left/right buttons, one can switch among the cities. The city switching should form a closed loop. For instance, if one keeps pressing only the right button, the cities should change as follows: Sydney -> Melbourne -> Brisbane -> Sydney -> Melbourne ->.... Pressing the left button follows similar pattern.

Once the weather data of a city is fetched, the corresponding Stage 2 page should never appear again. For example, the following is a possible interaction sequence:



2. *Technical Requirements:*

Create a GitHub repository to hold all your source code. There is no requirement on the structure of your project. However, a README.md file is required in the root folder of the repository and instructions on how to start your App should be given (No need to be in details. For example, a single command is enough).

Use application server **as both static file server and Restful API provider** (if you don't want to use Restful, you are free to choose other interfaces, such as GraphQL/SOAP).

First stage:

- This one-button static page should be served as the entry point of the App from the Backend. The preferred implementation is using some kind of template (view) engine, e.g. Pug, EJS, Mustache, NunJucks, etc for NodeJS.
  If you are not familiar with template engines, pure HTML is also acceptable.

Second/Third stages:

- The Frontend must be implemented via React.
- The related static files should be served from the Backend with a different URL from the First Stage.
- Conversion from Stage 2 to Stage 3 is required to be implemented by Ajax calls. Weather data should be fetched from the Backend API(s).
- Data of at least three cities are required. There is no limitation on the format, structure or contents of the mock data. The contents don't have to be the same as the example shown above.
- There is no requirement on the Restful API URL (or other interface you choose to use). Feel free to design how your data is provided by the Backend.

Important Notice:

Don't bother too much on the style and looking of the user interface. It is fine if you can not make it good-looking. You can even use plain html elements (e.g., button) without any style for the interaction. There are no penalties on the looking of the user interface.

Remember the logic flow and data manipulation are the top priorities.

<End of Task>

Hint:

For those who complete the above basic task and still have plenty of time left, please do one or more tasks from below. You will earn special attention from the recruitment team.

- Make the Web App responsive so that it is adaptive to different screen sizes.
- Demonstrate your documentation capability.
- Deployment:
  You can choose either option (you are welcome to do both)
  - Docker: build a docker image containing your production bundle, push it to DockerHub and submit the link to your image.
  - AWS: create a new account on AWS (or user your existing one if you have one) and use free-tier services to deploy your production bundle (suggestions: Elastic Beanstalk or simply an EC2 instance). Submit the link to your Web App (don't bother DNS and Routing).
- Database:
  Choose your favourite database (e.g. MongoDB, MySQL, Postgresql, etc.), insert your mock data into the database and emit a database query from your NodeJS Backend when the user interface asks for data from backend.
  If you choose to do this bonus point, leave all your database manipulation code in your repository, and point out which database you use in the README.md file.