TrueCaller - Assignment

Frontend

Truecaller Summer Internship Program 2019

Submitted by,
Abinaya Rajesh

April 23, 2019

Contents

1 Overview of the Project **3**

2 Design of the project **3**

2.1 Screen 1 3

2.2 Screen 2 5

3 Components **5**

3.1 index 5

3.2 Router 5

3.3 App 5

3.4 City Weather 5

3.5 Cards 6

3.6 CardUi 6

3.7 Precautions 6

3.8 DetailMap 6

3.9 Map 6

4 Conclusions **6**

1 Overview of the Project

This assignment, requires the create the web application that displays the temperature in three different cities namely Bangalore, Nairobi and Stockholm. Maximum freedom is left to the individual with respect to what design should be implemented.

A web application that displays temperature in Bangalore, Nairobi and Stockholm is created.

The first screen displays the below items:

- Name of the city
- Current temperature in the city
- Small map of the cities based on the coordinates
- Brief description about the city

The second screen gives more useful information about the weather as below:

- Current temperature
- Minimum temperature
- Maximum temperature
- Humidity
- Description
- Recommendations
- Detailed map of the city

2 Design of the project

This section summarizes all design aspects of the application. The application is built using the ReactJS. Two APIs were used to extract the location and the weather information as below,

- Maps Javascript API
- Weather API

2.1 Screen 1

- The current weather from different countries are fetched from the `WeatherAPI`.
- The maps based on the latitude and the longitude of the cities are also collected.
- The current weather is displayed with short description about the city and the map on the bootstrap cards with user experience in mind.
- Different visual effects like shades and animation has also been added to the cards.
- When you click on button on the cards or the map, you will be directed to the next screen which shows the detailed version of the map and the weather.
- The `Link` from the `react-router-dom` is utilized since the the state of the object is set by passing the key value arguments of the `Link`.

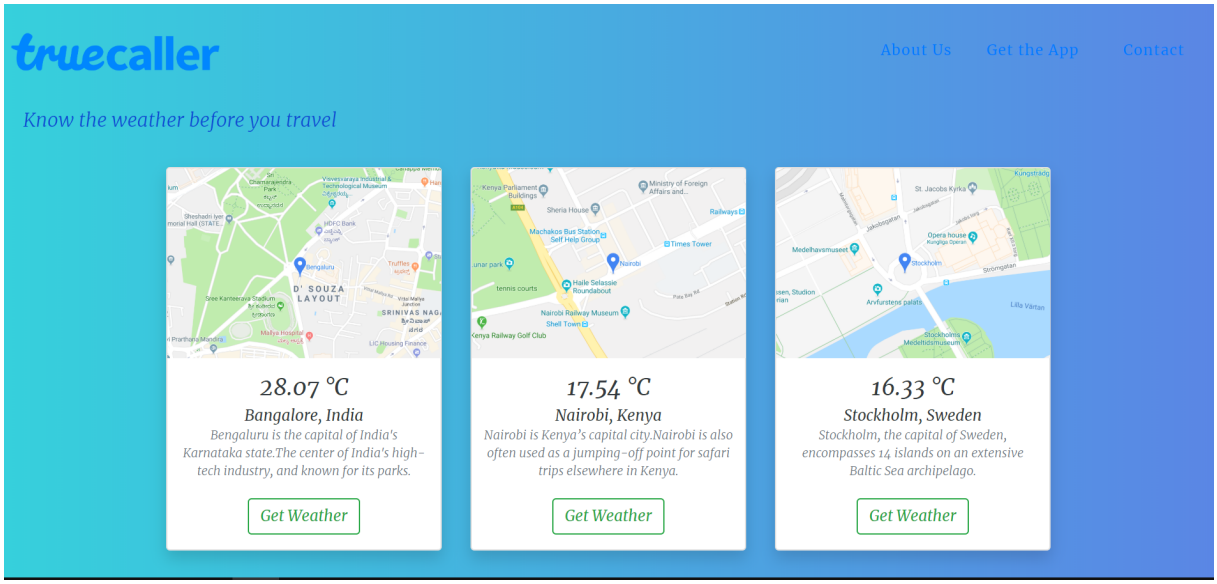

Figure 1: Screen 1

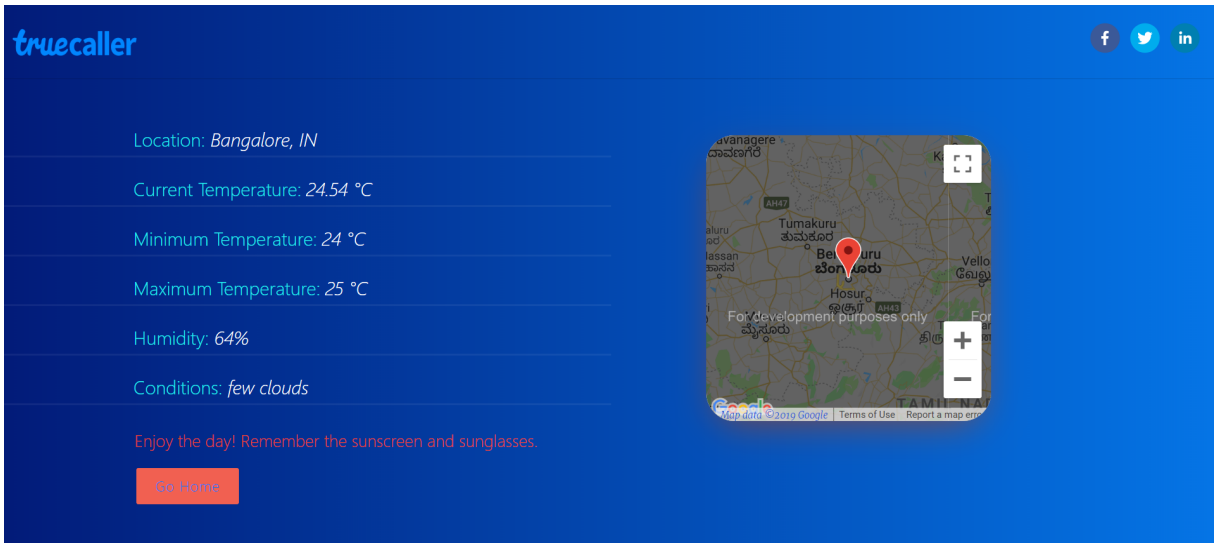

Figure 2: Screen 2

2.2 Screen 2

- The detailed weather and the map of different countries are fetched from the `WeatherAPI` and the `Google Javascript API`.
- The state is defined dynamically when the city name is clicked in the Screen 1 leading to the display of the detailed weather information and map of the particular city in Screen 2.
- The recommendation is given to the user based on the weather description that is fetched from the `main` attribute of `WeatherAPI` .
- The `Go Home` button and the `Truecaller` logo will navigate to the Screen 1 .
- The functionality to share the app via Twitter, Facebook and LinkedIn has been added by using `react share` .

3 Components

Some of the main technical aspects of the project have already been introduced in the previous sections, such as the `Link` and `Bootstrap cards`. This section will detail some other interesting aspects of the implementation and the different components involved in the development of the application.

3.1 index

The `index` component is the root and all the other components are rendered out through this component. The `ReactDOM` is used in this project to render out the components.

3.2 Router

The `Router` component as the name suggests, it is used to route the components to different paths ie; to the home page and the detail page. It has two components `App` and `CityWeather`.

3.3 App

The `App` component represents the first page of the application which encloses the `Card` component. It has a class which renders out the weather information in Bootstrap Cards through `Card` tag.

3.4 City Weather

- In `CityWeather` is a react component, the weather details are rendered out form the WeatherAPI. A state object is created with different attributes like temperature, lowest temp, highest temp, city, country, humidity, description, main, lat, and long is created. This state is set only when the details are fetched from the Weather API passing the value of the city and the API Key in the API link.
- The extracted values are displayed on the interface clearly by using the style sheets. The values fetched are always checked and then displayed without it may result in missing data on the display. For example, `this.state.city` is the expression that is used to check whether the city has fetched the value from the API.
- Secondly, the values of the attributes lat(latitude) and the lon(longitude) is fetched dynamically from the API and passed on to the `DetailedMap` component that will in turn assist in extract the detailed map. The states of the attributes `main` is passed to the `Precautions` tag. The button tag is also incorporated to direct to the `Go Home` button.

3.5 Cards

- The main purpose of the `Card` JSX element is to display the detailed weather info with the small map on Bootstrap cards. The screen is split equally by col-md-4 class and the cards are displayed in parallel. The temperature is fetched from the API by using the city names and displayed on the cards.
- A button is also added on the card to direct to the next page. The button enclose a `Link` tag from the `react-router-dom` which will pass the clicked city name to set the state of the object to help in retrieving the weather information dynamically.

3.6 CardUi

- The `CardUi` is a JSX component which basically designs the look of the card by utilizing different classes in `Bootstrap Cards 4`.
- The props are passed through different tags and are controlled by the `Cards` component.

3.7 Precautions

The Precautions is a reactjs component that renders the conditions of the weather from the cityWeather component and returns the recommendation to the user based on `main` which represents the main description of the weather.

3.8 DetailMap

The `DetailMap` component renders the *latitude* and the *longitude* from the `CityWeather` component and returns the map which is fetched from `Maps` component. It represents the detailed map. This will also place a marker on the city based on the coordinates by using `google-maps-react`.

3.9 Map

The `Map` component extracts the detailed maps from the *Google JavaScript API*. The function `componentDidMount()` lifecycle method is used to carefully invoke the map when the component is mounted.

4 Conclusions

The project was completed successfully, and I had fun in performing this interesting assignment. From the learning perspective, many are the key takeaways of this project, related to fetching the right values from the API. Furthermore, fetching the value from the WeatherAPI was easier compared to adding Marker to the Google API.