Weather-app-forecast
code review

1). Navbar.js

```
/*
  A semicolon is missing at the end of the import statement.
*/
1 import React, { Component } from 'react'
```

```
/*
  This could be expanded on multiple lines to inprove readability:

  ({
    menu: !this.state.menu
  });
*/
10 this.setState({ menu: !this.state.menu })
```

```
/*
  In JS it is good practice to use single quotes '' and only use
double quotes inside single quotes if needed and also for HTML/
JSX.
*/
16 const show = (this.state.menu) ? "show" : "" ;
```

```
/*
  Why is there a space between the tag start and end?

  <nav></nav>
*/
21 < nav className="navbar navbar-expand-md bg-dark navbar-dark
rounded-top" >
```

```
/*
  This line is way too long, in our code we usually split lines
after 100 characters.
  Splitting example for JSX/HTML tags:
```

```
  <button
    className=“navbar-toggler”
    type=“button”
    data-target=“#collapsibleNavbar”
  >
*/
27 <button className="navbar-toggler" type="button"
onClick={this.toggleMenu} data-toggle="collapse" data-
target="#collapsibleNavbar">
```

```
/*
  The below can also be written using template literals:

  className={`collapse navbar-collapse ${show}`}
*/
33 <div className={"collapse navbar-collapse " + show }>
```

```
/*
  No space in the nav close tag.
  </nav>
*/
47 </nav >
```

```
/*
  The below misses a semicolon, as it ends the return statement:

  );
*/
)
```

```
/*
  The below misses a semicolon.
*/
52 export default Navbar
```

2). AddCardAuto1.js

```
/*
  All the JS code needs to use single quotes.
  Double quotes are only used when you want to quote something
inside a string, in HTML or JSX.
  You can fix this across all files.
*/
1 import React, { Component } from "react";
2 import Script from "react-load-script";
```

```
/*
  The class name here is not suggestive enough.
*/
5 export default class AddCardAuto1 extends Component {
```

```
/*
  The check below will error if there is no data[0].
  A safer check here would be:

  if (data[0] && data[0].WeatherText) {}
*/
35 if (data[0].WeatherText) {
```

```
/*
  Instead of targetting the HTML element and changing its value,
  you can just change this.state.query, as you're reading that
into the "value" property of the input on L129. When you update
the state value, the component will rerender and your input will
display the new state.
  I see you're using undefined as default value - is there any
specific reason for this? If not, the default value could be the
placeholder itself, which you can then change with '':
  1). Default this.state.query is your placeholder, e.g. "Please
enter a city";
  2). On focus, if this.state.query is the placeholder, then
change this.state.query with '' so that user can input their own
text, otherwise don't do anything;
  3). On blur, if there is no user text in the user input, change
this.state.query back to the placeholder, otherwise don't do
anything.
*/
58 document.getElementById("autocomplete").value = "";
```

```
/*
  From what I see below, address is an array with multiple address
objects.
  In this case, maybe a variable name in plural is better as it’s
more suggestive:
  addresses, arrAddresses

  Also, you need to rewrite this check, as you are using
address[0].long_name below.
  If there is no address[0], trying to access a property on it
will result in an error.

  if (address.length) {}
*/
if (address) {
```

```
/*
  Naming for the below isn't consistent with the rest of the code,
  which is written in camelCase.
*/
const api_call = await fetch(url);
```

```
/*
  The variables below can be declared with let instead of var, as
you only need them in this particular block statement.
  Also, can the photos array be empty? If yes, then you need to
add a check here, as accessing getUrl() on a non-existent array
item will result in an error:

  let picUrl = (photos[0]) ? photos[0].getUrl() : defaultPic;
*/
var photos = addressObject.photos;
var picUrl = photos[0].getUrl();
```

```
/*
  Is there any specific reason for which you dynamically load the
script here?
  If you don't need a specific reason, you can simply load it in
the HTML file and then ditch react-load-script entirely.
*/
```

```
<Script
  url="https://maps.googleapis.com/maps/api/js?key=AIzaSyBp4lsvgALACqdsxsEnW6A4y2e8CP3F9SU&libraries=places"
  onLoad={this.handleScriptLoad}
/>
```

```
/*
  The style below can be moved in a separate js file as an object
  and then referenced here - I can show you how.
*/
<input
  type="search"
  className="form-control"
  id="autocomplete"
  placeholder=""
  value={this.state.query}
  onFocus={this.handleFocus}
  style={{
    margin: "0 auto",
    maxWidth: 800
  }}
/>
```

3). Card1.js

```
/*
  The below can be written using a template literal:
  `https://vortex.accuweather.com/adc2010/images/slate/icons/${this.props.city.icon}.svg`
*/
const icon_url = 'https://vortex.accuweather.com/adc2010/images/slate/icons/' + this.props.city.icon + ‘.svg';
```

```
/*
  this.props.city below seems to refer to an object, as it's got multiple property keys on it.
  Since that is the case, a better naming would be appropiate, e.g. objCity or cityObject - and then it makes sense to access this.props.objCity.city.
*/
```

```
36 const id = this.firstWord(this.props.city.city);
```

```
/*
  Don't forget about single quotes vs double quotes - in all
files.
*/
37 const panelRef = "#" + id;
```

```
/*
  I like the usage of an object in order to declare styles - as
per recommendation in the AddCardAuto1 file - if you want, these
can be kept in a separate JS file and imported at the top.
*/
38 const imageBGstyle = {
39   backgroundImage: `url('${this.props.city.image}')`,
40   // filter: `brightness(50%)`
41 };
```

```
/*
  For the below code in the return statement:
    - Solve spacing and formatting issues;
    - Don't forget to expand lines over 100 characters
    - Semicolon missing at the end of the return statement
*/
46 return ( ... );
```

4). App1.js

```
/*
  Single quotes vs double quotes on the lines below (L17, L19) and
throughout the file.
*/
17 mode: “edit",
18 editMode: false,
19 buttonClass: "btn btn-primary align-items-center addButton fa
fa-3 fa-info-circle"
```

```
/*
  You need a semicolon on L27.
*/
```

```
25 this.setState({
26   show: val
27 })
```

```
/*
  Nice! You can keep the value on the same line, though, as it's
quite readable anyway.
*/
33 cards:
34   [...this.state.cards, val],
```

```
/*
  The return statement needs a semicolon on L48.
*/
44 return {
45   cards: state.cards.filter(
46     card => card.id !== id
47   )
48 }
```

```
/*
  Indent can be reduced by one level below just by removing the
else statement entirely and adding the code inside below, after
you add a return statement in the if body:

  if (something) {
    // Do something
    return; // <- This prevents executing the code below the if
statement
  }
  // Do some other thing that you would put in an else body
*/
78 if (this.editMode) {
79   console.log('Editmode true');
80   this.setState({
81     // mode: "done",
82     editMode: true,
83     buttonClass: base + done,
84     // showInput: true
85   });
86 } else {
87   console.log('Editmode false');
88   this.setState({
89     editMode: false,
90     buttonClass: base + fade + edit,
91     // showInput: false
92   });
93 }
```

```
/*
  The lines below, but also L118 and L119 need to be expanded as
they exceed 100 characters.
*/
111 <div className="col-md-6 flex-grow-1 d-flex align-items-center
justify-content-end w-75 bg-info">
112 <a className={this.state.buttonClass} id={this.state.mode}
value="add" onClick={this.handleClick} href="#"></a>
```

5). helpers.js

```
/*
  Same comment about the naming of api_call being inconsistent
with the camel casing conventions for this particular variable
type.
*/
7 const api_call = await fetch(url);
```

```
/*
  Same thing about accessing a property on a *possibly* inexistent
object in an array.
  The below condition needs to be checked for the existance of
data[0] first, and, if data[0] can hold multiple values, you also
need to check whether data[0] is an Object.

  Regarding the if/else conditionals, you can apply an early
return in order to improve code readability:

  if (!data[0] || !data[0].WeatherText) {
    // Send error to the user
    return; // <- prevents running the code below
  }
  this.props.addCity({
    // whatever goes here
  });
*/
if (data[0].WeatherText) {
  // ...
} else {
  console.error("Error: with choice of City or Country");
}
```