Web App Design with React Week 15-Ayushjav- Coding Assignment

Points possible: 75

URL to GitHub Repository:

Instructions: In VS Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. Create a new repository on GitHub for this week's assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week's repository to this document where instructed and submit this document to your instructor when complete.

Coding Steps:

1. Using the Houses API, or any open API of your choice you can find online, create a single page that allows for all 4 crud operations to be performed on a resource from that API. Create a React component (or more, if needed) to represent the resource. Make all forms and other necessary UI pieces their own components as reasonable.

Screenshots of Code:

Components/House.js

PROMINEO TECH

House.js

src > components > House.js > ...

```
1   import React from "react";  6.9k (gzipped: 2.7k)
2   import { NewRoomForm } from "./NewRoomForm";
3
4   //functional component
5   export const House = props => {
6     //deconstruct the props to get an update -- house, all the data that reps the
      house, updateHouse, the method to update it
7     const { house, updateHouse } = props;
8
9     const deleteRoom = roomId => {
10      //when deleting a room, updating a house, the updated house to the results that
        come back when filter out the room that has a matching id. Will use filter method
        on the array.
11      const updatedHouse = {
12        //using spread
13        ...house,
14        rooms: house.rooms.filter(x => x._id !== roomId),
15      };
16      updateHouse(updatedHouse);
17    };
18
19    //new method to add a room
20    const addNewRoom = room =>
21      //create a new array using spread. Insteading a pushing to rooms list, want to
        get a new object/array, not modify it. Create a new array, spread from the
        houses, and adding the new room -- takes all the values from the old array and
        adds a new room to it. Removed the return since it will automatically returned bc
        we used the fat arrow.
22      updateHouse({ ...house, rooms: [...house.rooms, room] });
23
24    //create component within this component for rooms. Smaller component, and are
      separated logically as own, but only related to the house, so keeping it here.
25
26    //function to check if the room name is null or empty
27    const checkRoomName = room => {
28      console.log("checkRoomName", room);
29      if (room === null || room === "") {
30        return "";
31      } else {
32        return room.name;
33      }
34    };
35
36    //function to check if the room area is null or empty
37    const checkRoomArea = room => {
38      //If the room area is null or empty return an empty string
39      if (room === null || room === "") {
40        return "";
41      }
42      //If the room area is not null or empty return the room area
43      else {
44        return room.area;
45      }
46    };
47
48    //plural (rooms) -- this component is the rooms, not each individual room and this
      component will create an instanace of rooms as an li
49    const rooms = () => (
50      <ul className='homedescription'>
51        {house.rooms.map((room, index) => (
52          <ul key={index}>
53            {console.log(house.rooms)}
54            <label className='room'>{`${checkRoomName(room)}`}</label> <br />
55            <label className='area'>
56              {`Area: ${checkRoomArea(room)} square feet`}
57            </label>
```

```
58                <button
59                  className='btn
60                  onClick={e => deleteRoom(room._id)}
61                >
62                  Delete
63                </button>
64              </ul>
65            ))}
66          </ul>
67        );
68
69        return (
70          <div className='housename'>
71            <h3> Name of home: </h3>
72            <div className='housename-name'>
73              <h4>{house.name}</h4>
74            </div>
75
76            {
77              //rooms is a function, passing props (what's inside the ({})). The props are
                all the rooms, the house id, and deleteRoom
78              rooms({ rooms, houseId: house._id, deleteRoom })
79            }
80            <NewRoomForm addNewRoom={addNewRoom} />
81          </div>
82        );
83      };
84
```

const deleteRoom: (roomId: any) => void

Components/HouseList.js

PROMINEO TECH

JS HousesList.js ×

src > components > JS HousesList.js > ...

```
1   import React from "react";  6.9k (gzipped: 2.7k)
2   import { House } from "./House";
3   import { houseApi, housesApi } from "../rest/HousesApi";
4   import { NewHouse } from "./NewHouse";
5
6   //making a class-based component
7   export class HousesList extends React.Component {
8     //dont need a constructor for the state
9     state = {
10      //will start as empty array
11      houses: [],
12    };
13
14    componentDidMount() {
15      this.fetchHouses();
16    }
17    fetchHouses = async () => {
18      //using the GET method to make a request for the data
19      const houses = await housesApi.get();
20      //this sets the state of the houses array
21      this.setState({ houses });
22    };
23
24    updateHouse = async updatedHouse => {
25      await housesApi.put(updatedHouse);
26      //calls the this.setState method to update the state of the houses (the empty
        array) after the current houses has been updated
27      this.fetchHouses();
28    };
29
30    render() {
31      return (
32        //className to do CSS styling
33        <div className='houselistmain'>
34          <div className='newhousearea'>
35            <NewHouse />
36          </div>
37
38          <div className='house-list'>
39            <h2> List of home</h2>
40            {this.state.houses.map(house => (
41              <House
42                house={house}
43                key={house._id}
44                updateHouse={this.updateHouse}
45              />
46            ))}
47          </div>
48        </div>
49      );
50    }
51  }
52
```

Components/Navbar.js

Navbar.js

src > components > JS Navbar.js > ...

```
1   import React, { Component } from "react";  6.9k (gzipped: 2.7k)
2
3   export default class Navbar extends Component {
4     render() {
5       return (
6         <nav className='navbar navbar-expand-lg'>
7           <div className='container-fluid'>
8             <a className='navbar-brand' href='#'>
9               SunScreen Realty LLC
10            </a>
11            <button
12              className='navbar-toggler'
13              type='button'
14              data-bs-toggle='collapse'
15              data-bs-target='#navbarNav'
16              aria-controls='navbarNav'
17              aria-expanded='false'
18              aria-label='Toggle navigation'
19            >
20              <span className='navbar-toggler-icon'></span>
21            </button>
22            <div className='collapse navbar-collapse' id='navbarNav'>
23              <ul className='navbar-nav'>
24                <li className='nav-item'>
25                  <a className='nav-link' aria-current='page' href='#'>
26                    Home
27                  </a>
28                </li>
29              </ul>
30            </div>
31          </div>
32        </nav>
33      );
34    }
35  }
36
```

Components/Navbar.js

NewHouse.js

src > components > NewHouse.js > ...

```
1  import React, { useState } from "react";  6.9k (gzipped: 2.7k)
2  import { houseApi, housesApi } from "../rest/HousesApi";
3
4  //functional component that takes props
5  export const NewHouse = () => {
6    //use hooks,
7    const [name, setName] = useState("");
8
9    //need to define onSubmit
10   const onSubmit = e => {
11     e.preventDefault();
12     //if name and area are true, have values
13     console.log(name);
14     housesApi.post(name);
15     // e.target.value;
16     setName("");
17   };
18   //create the JSX for the return value
19   return (
20     <div className='newhouse'>
21       <h2> Add a new House</h2>
22       {/* creating a form */}
23       <form onSubmit={onSubmit}>
24         {/* when the text on the input changes, calling the setName method and
           setting it to the target value*/}
25         <input
26           type='text'
27           placeholder='Name of your new house'
28           onChange={e => setName(e.target.value)}
29           value={name}
30           name={name}
31         />
32         <button className='addRoom' type='submit'>
33           Add House
34         </button>
35       </form>
36     </div>
37   );
38 };
39
```

Components/NewRoomForm.js

PROMINEO TECH

NewRoomForm.js

src > components > NewRoomForm.js > ...

```
import React, { useState } from "react";  6.9k (gzipped: 2.7k)

//functional component that takes props
export const NewRoomForm = props => {
  //use hooks
  const [ const area: undefined te("");
  const [area, setArea] = useState(undefined);

  //since area a number, validation so that it is a number
  const handleAreaInput = e => {
    //using parse to make sure it's an interger and round to 10
    const int = parseInt(e.target.value, 10);
    //if Area greater or = to 0 (? means true), then pass in. If not (:), then empty
    string
    setArea(int >= 0 ? int : "");
  };

  //need to define onSubmit
  const onSubmit = e => {
    e.preventDefault();
    //if name and area are true, have values
    if (name && area) {
      //calling addNewRoom method and passing in an object with name and area
      props.addNewRoom({ name, area });
      //once passed, setting it back to empty
      setName("");
      setArea("");
    } else {
      console.log("invalid input");
    }
  };
  //create the JSX for the return value
  return (
    <div>
      <h5> Add a new room</h5>
      {/* creating a form */}
      <form onSubmit={onSubmit}>
        {/* when the text on the input changes, calling the setName method and
        setting it to the target value*/}
        <input
          type='text'
          placeholder='Room Name'
          onChange={e => setName(e.target.value)}
          value={name}
        />
        <br />
        <input
          type='text'
          placeholder='Area in square feet'
          onChange={handleAreaInput}
          value={area}
        />
        <button className='addRoom' type='submit'>
          Add Room
        </button>
      </form>
    </div>
  );
};

```

PROMINEO TECH

Rest/HousesApi.js

PROMINEO TECH

JS HousesApi.js

src > rest > JS HousesApi.js > ...

```
//Where logic to make network call for the Houses_ENDPOINT. It's own component so that other components can use it when need to make an API call. (in other method, had it tied to the App.js component) - Reusable

//The two methods used to get request and update request (update = add/delete new rooms)

const HOUSES_ENDPOINT = "https://ancient-taiga-31359.herokuapp.com/api/houses";

//normal class to house a functions/methods for fetch.
class HousesApi {
  get = async () => {
    //using a try/catch in case something goes wrong
    try {
      //makes the call for data
      const resp = await fetch(HOUSES_ENDPOINT);
      //have data returned into json
      const data = await resp.json();
      if (data.length > 3) {
        data.reverse().length = 3;
      }
      return data;
      //(3) is the exception, what went wrong
    } catch (e) {
      console.log("Ooops, fetch is being a bitch", e);
    }
  };
  //method for PUT (Update), takes a parameter, House, to update it
  put = async house => {
    try {
      //want the id from the house to identify it from the other houses in the array, with 2nd object from the request data
      const resp = await fetch(`${HOUSES_ENDPOINT}/${house._id}`, {
        method: "PUT",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(house),
      });
      return await resp.json();
    } catch (e) {
      console.log("Ooops, looks like updating the houses has an issue", e);
    }
  };

  post = async housename => {
    try {
      //want the id from the house to identify it from the other houses in the array, with 2nd object from the request data
      const resp = await fetch(`${HOUSES_ENDPOINT}`, {
        method: "POST",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify({ name: housename }),
      });
      return await resp.json();
    } catch (e) {
      console.log("Ooops, looks like updating the houses has an issue", e);
    }
  };
}

//putting the HousesApi method into another variable to use in other components.
export const housesApi = new HousesApi();
```

App.js

Index.css

PROMINEO TECH

index.css

src > # index.css > ...

```css
/* general css */

* {
  top: 0;
  padding: 0;
  box-sizing: border-box;
}

.body {
  font-family: Arial, Helvetica, sans-serif;
  background: #333;
  color: #93cb52;
}

/*css for navbar*/
.navbar {
  background: #7ab436;
  color: #fff;
}

/* css for the main header */
header {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  text-align: center;
  height: 80%;
}

header h1 {
  text-align: center;
  color: #def50f;
  border-bottom: solid #24ac52;
  padding: 0.75rem 1.5rem;
}

/*css for the main house list*/
main {
  width: 80%;
  max-width: 960px;
  margin: 1rem auto;
}

/*new house area on the house list*/
.newhouse {
  text-align: center;
  padding: 30px;
}

/*List of homes*/

.house-list h2 {
  color: white;
  padding: 15px 0px;
  border-bottom: white solid;
  text-align: center;
}

.housename h3 {
  color: rgb(101, 213, 22);
}

.housename-name h4 {
  padding: 0px 30px;
  color: #e5e8e0;
```

```
  color: #e5e8e0;
  font-size: 40px;
}

.homedescription label {
  padding: 0px 15px;
}

.homedescription .room {
  font-size: 25px;
  font-style: normal;
}

.homedescription .area {
  padding: 0px 30px;
}

.addRoom {
  background-color: rgb(195, 240, 162);
  border: solid;
}

.addRoom:hover {
  background-color: yellow;
  opacity: 0.4;
}

```

Screenshots of Running Application:

PROMINEO TECH