Lab: Content Rating Application to Like or Dislike Content

Estimated time needed: 40 minutes

What you will learn

In this lab, you will create a React component called 'ContentRating' where the component will let users rate material by clicking 'like' or 'dislike' buttons. When the component is rendered for the first time, both the like and dislike counts are set to zero. You will create a method that changes the state to add one to the number of likes when a user selects the 'like' button. In the same way, clicking the 'dislike' button, the method will add one to the number of dislikes. This action lets users rate the content in an interactive way using like or dislike button and provides feedback to content creators.

Learning objectives

After completing this lab, you will be able to:

- Handle the component state in React using state variables to keep track of the number of likes and dislikes
- Execute event handling when a user selects a button to like or dislike content
- Create components and reuse them by building a Toggle component that includes both the content and rating features
- Create an interactive user interface with React that can handle state and user events and render complex UI elements

Prerequisites

- Basic knowledge of HTML
- Intermediate knowledge of JavaScript
- Basic knowledge of react class component and state management

Step 1: Setting up the environment

1. From the menu on top of the lab, click the **Terminal** tab at the top-right of the window shown at number 1 in the given screenshot, and then click **New Terminal** as shown at number 2.

2. Now, write the following command in the terminal to clone the boiler template for this React application and select Enter.

```
git clone https://github.com/ibm-developer-skills-network/content_rating.git
```

3. The above command will create a folder, "content_rating" under the "Project" folder, and you can see the structure in the screenshot. This includes the class component named "CourseRating.jsx" and a CSS file named "CourseRating.css.“

4. Next, you need to go inside the "content_rating" folder in the path of the terminal. For this you need to write the given command in the terminal. This action will set your terminal path to run the React application within the content_rating folder.

```
cd content_rating
```

5. To ensure the code you have cloned is working correctly, you need to perform the following steps:

    - Write the given command in the terminal and hit Enter. This command will install all the necessary packages to execute the application.

```
npm install
```

    - Then, perform the following command to run the application, providing you with port number 4173.

```
npm run preview
```

6. To view your React application, click the Skills Network icon on the left (refer to number 1). This action will open the **SKILLS NETWORK TOOLBOX**. Next, click **Launch Application** (refer to number 2). Enter the port number **4173** in **Application Port** (refer to number 3) and click .

7. The output will display as shown in the given screenshot.

8. You can preserve your latest work on this lab by adding, committing, and pushing it to your GitHub repository. This ensures that even if you're not working on the task continuously, your progress will be saved, allowing you to resume from where you left off.

Note: Step 8 is optional.

Step 2: Setting the initial state

1. Next open the rating component by navigating to the **ContentRating.jsx** component located in the **Components** folder of the **src** directory in your cloned **content_rating** folder.

2. The basic structure of this component will be as shown in the screenshot.

3. You need to initialize the states for the likes and dislikes count within the constructor of the current component after **super()** method. Create **this.state** object and initialize the values of likes and dislikes to `0` for the initial state.

```
this.state = {
     likes: 0,
     dislikes: 0
   };
```

4. You can remove the `<h1>Text Content Rating</h1>` element and create the `div` tag with class name **content-rating** within `<>...</>` under `return` of class component with the help of given command.

```
<div className='content-rating'></div>
```

   This `div` will act as the parent div for other tags. Include given command inside the fragmenets.

5. Then create a `<p> Add text here</p>` tag inside `div` tag with some content relatede to any topic that users can like or dislike.

```
<div className='content-rating'>
    <p>
    //Add text here
    </p>
</div>
```

6. Create one more `<div>` tag with the class name **rating-buttons** after the paragraph tag inside the parent div tag.

```
<div className='content-rating'>
    <p>
    //Add text here
    </p>
    <div className='rating-buttons'></div>
</div>
```

7. Create two buttons inside the div with class name **rating-buttons**. One for like and another for dislike. Then, display the values of the variables as text within these buttons, which you have initialized under the `this.state` object.

```
<div className='content-rating'>
    <p>
    ---Add text here---
    </p>
    <div className='rating-buttons'>
    <button className="like-button">
        Like ({this.state.likes})
      </button>
      <button className="dislike-button">
        Dislike ({this.state.dislikes})
      </button>
    </div>
</div>
```

Step 3: Create event handling

1. In this step, you will create events that will handle the clicks performed on the likes and dislikes buttons.

2. Create two event handlers under the `this.state` object with the name **handleLike** for the like button and **handleDislike** for the dislike button.

```
constructor() {
   super();
   this.state = {
     likes: 0,
     dislikes: 0,
     handleLike:() => {
     },
     handleDislike:() => {
     }
 }
   }
```

3. Now, in these two events handlers write the code logic to increase and decrease the likes and dislikes button values.

```
constructor() {
   super();
   this.state = {
     likes: 0,
     dislikes: 0,
     handleLike:() => {
       this.setState((prevState) => ({
         likes: prevState.likes + 1
       }));
     },
     handleDislike:() => {
       this.setState((prevState) => ({
         dislikes: prevState.dislikes + 1
       }));
     }
 }
   }
```

   - In the above code, the arrow functions, handleLike and handleDislike, are called when the user clicks the "like" or "dislike" button, respectively.
   - The functions utilize the setState method to update the component's state. Inside setState, the previous state (prevState) is accessed, which holds the previous state of the component before the update.
   - Then the likes or dislikes count is incremented by one and set as the new state value.
   - This ensures that each button click accurately updates the corresponding count in the component's state.

Step 4: Call event handlers

1. Now you need to call these event handlers using the click of like and dislike button.

2. For this, you need to use the `onClick` event on buttons and call **handleLike** and **handleDislike** on **Likes** and **Dislikes** buttons, respectively, as shown in the given code.

```
<button className="like-button" onClick={this.state.handleLike}>
           Like ({this.state.likes})
```

```
                </button>
                <button className="dislike-button" onClick={this.state.handleDislike}>
                  Dislike ({this.state.dislikes})
        </button>
```

Step 5: Check the output

1. Now run the stop the execution of the React application in the terminal by performing `ctrl+c` to quit.

2. Then, write the given command in the terminal and hit Enter.

   ```
   npm run preview
   ```

3. To view your React application, refresh the already opened webpage for the React application on your browser. If it is not open, then click the Skills Network icon on the left panel. This action will open the "SKILLS NETWORK TOOLBOX." Next, select "Launch Application". Enter the port number **4173** in "Application Port" and click .

4. The output will display as per the given screenshot.

   This screenshot shows content of React library. You can write your own content within `<p>` tag in place of placeholder `//Add text here`. Your output will be based on the text that you have added in place of `//Add text here` placeholder.

5. Check the functionality of likes and dislikes by clicking the buttons. If you will click 5 times on **like** button and three times on **dislike** button then it will be seen according to given screenshot.

6. This represents that this text gets 5 like and 3 dislikes.

7. ▶ Click here to see the full sample solution for "ContentRating.jsx"

8. ▶ Click here to see the full sample solution for "App.jsx" parent component.

Step 6: Practice Exercise

1. In this exercise you need to create logic to calulcate total number of ratings combining both likes and dislikes within `ContentRating.jsx` component itself.

2. For this you need to declare one more variable named `totalRatings` inside `this.state` and initialize it with 0.

3. Now create logic to calulatetotal number of likes and dislikes for the content and store total number in variable `totalRatings`.

   Hint: Include `totalRatings` varibale in both likes and dislikes functions while increments the total number of ratings.

▶ Click here for the answer

4. Now display the total ratings iside jsx synatx after buttons tag.
   Hint: Use `{}` to display the total ratings variable using **this.state**.

▶ Click here for the answer

5. Check the output by re-running the application in terminal again.

6. The output, as shown in the provided screenshot, will display the total number of ratings as 9 when there are 5 likes and 4 dislikes. Clicking on either the like or dislike button will increment the total ratings accordingly.

7. ▶ Click here to see the full sample solution for "ContentRating.jsx"

Note:– To see the latest changes, you need to execute `npm run preview` again in the terminal.

Congratulations! You have created your second React application to give rating to content!

Conclusion

- In this lab, you have learned how to set up and create state variables in the constructor of a React class component. The state variables let the component handle and keep track of dynamic data like "likes" and "dislikes".
- You have understood how to make arrow functions in the state of a component, which contains the code for changing certain state variables when the user does something like click a button.
- Using event handling, you have learned how to connect these arrow functions to the onClick events on related UI elements in React.The functions let the state change dynamically based on user interaction.
- You have also acquired the information to add dynamic state data to the rendered UI using the JSX markup. This shows the current counts of likes and dislikes along with descriptive content, giving users a complete way to rate content.

Author(s)

Richa Arora