实验：为调查创建反馈表单

预计所需时间: 40分钟

您将学到什么

在本实验中，您将使用 React 函数组件创建一个反馈表单，并使用 **useState** 钩子管理用户详细信息。您将实现事件处理程序来管理表单输入更改，验证用户输入，并处理表单提交。此外，您将使用 **confirm** 方法创建一个确认对话框，以在最终提交之前确认用户详细信息。成功提交后，您将重置表单字段并向用户显示感谢信息。本实验将为您提供在 React 应用程序中构建交互式表单和处理用户输入的实际经验。

学习目标

完成本实验后，您将能够：

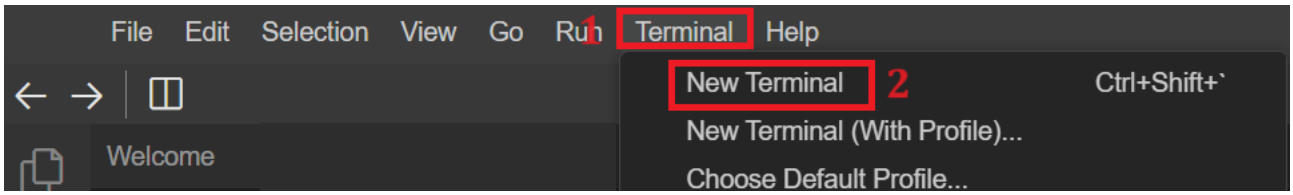
- 创建一个表单以收集用户反馈，包括他们的姓名、电子邮件和反馈信息。
- 处理表单提交，包括数据验证，以确保用户在提交表单之前输入他们的姓名并提供反馈。
- 在用户提交反馈后向他们显示一个确认对话框，显示他们输入的信息并提示他们在最终提交之前进行确认。
- 重置表单字段以清除输入值，并为用户提供一个干净的表单以提交额外的反馈。

先决条件

- 基础的 HTML 知识
- 中级的 JavaScript 知识
- 基础的 React 函数组件、useState 钩子和表单处理知识

步骤 1：设置环境

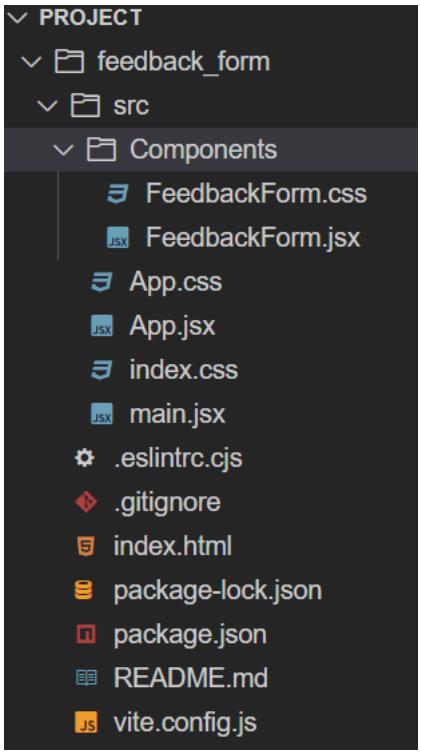
1. 在实验室顶部的菜单中，点击窗口右上角标记为 1 的 **终端** 选项卡，然后点击标记为 2 的 **新建终端**。



2. 在终端中输入以下命令并按 Enter 以克隆此 React 应用程序的模板。

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

3. 这将创建一个名为 **feedback_form** 的文件夹在项目文件夹下，结构将如给定的截图所示。feedback_form 应用程序包含一个名为 **FeedbackForm.jsx** 的类组件和一个名为 **FeedbackForm.css** 的 css 文件。



4. 在终端中输入命令以进入 **feedback_form** 文件夹。该命令将设置您的终端路径，以便在 **feedback_form** 文件夹中运行 React 应用程序。

    ```
    cd feedback_form
    ```

5. 为了确保您克隆的代码正常工作，您需要遵循以下步骤：

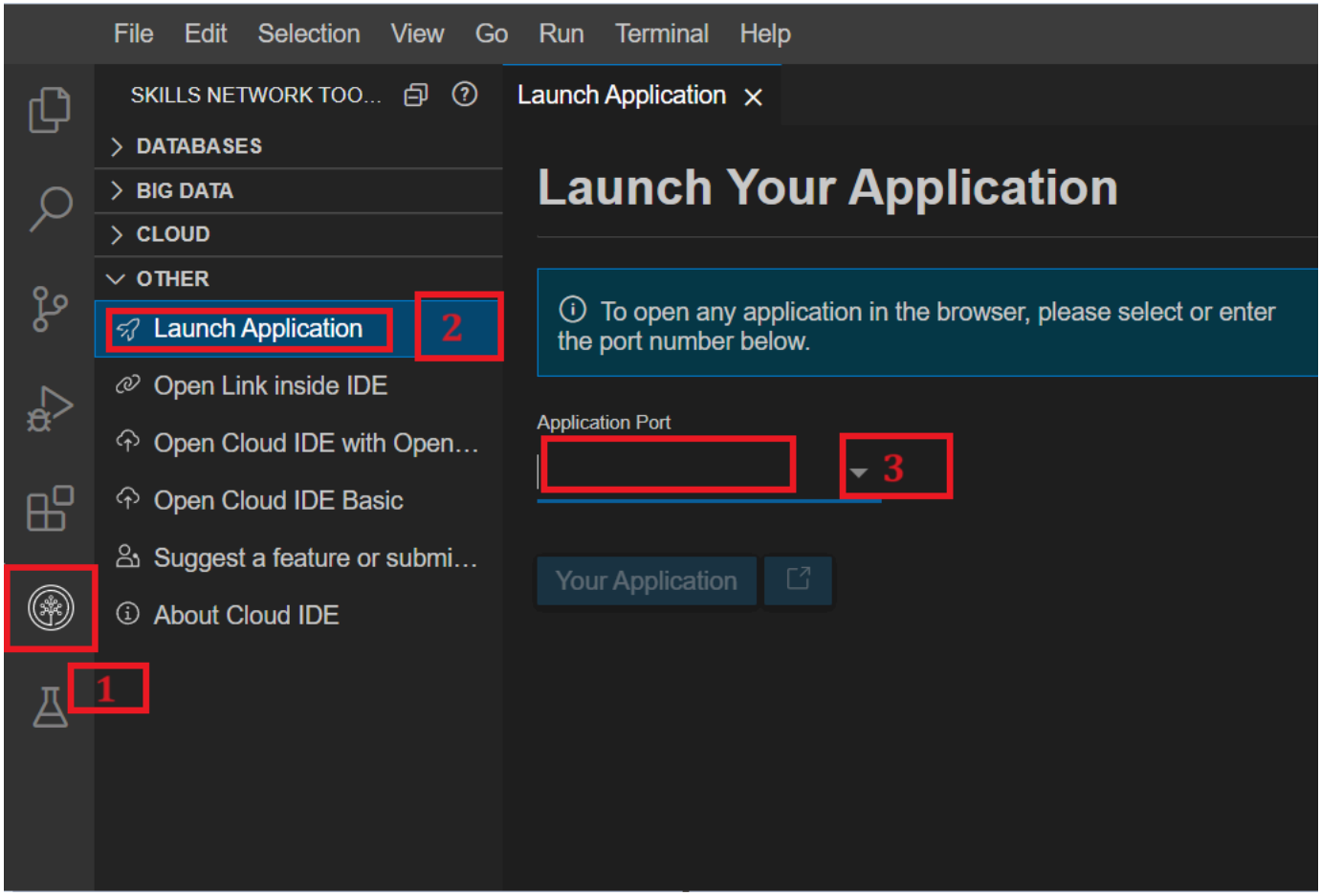
    - 在终端中输入以下命令并按 Enter 安装所有必要的软件包以执行应用程序。

        ```
        npm install
        ```

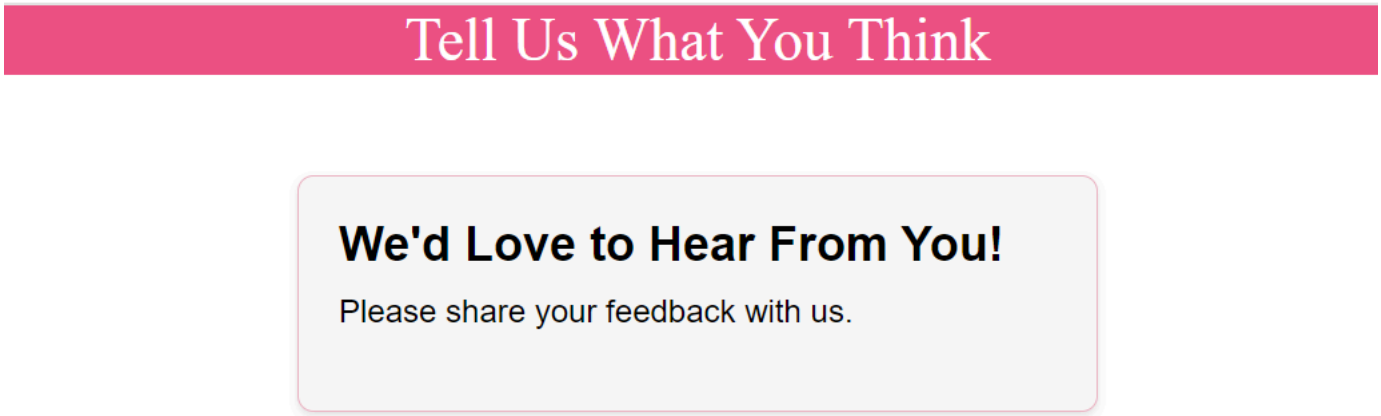
    - 然后执行以下命令以运行应用程序，并提供端口号 4173。

        ```
        npm run preview
        ```

6. 要查看您的 React 应用程序，请点击左侧面板上的技能网络图标（参见编号 1）。此操作将打开 **Skills Network Toolbox**。接下来，点击 **Launch Application**（参见编号 2）。在 **Application Port** 中输入端口号 **4173**（参见编号 3），然后点击 。



7. 输出将显示如给定的截图所示。



8. 您可以通过添加、提交和推送到您的 GitHub 仓库来保存您在此实验室的最新工作。这确保即使您没有持续进行任务，您的进度也将被保存，允许您从中断的地方恢复。
注意：步骤 8 是可选的。

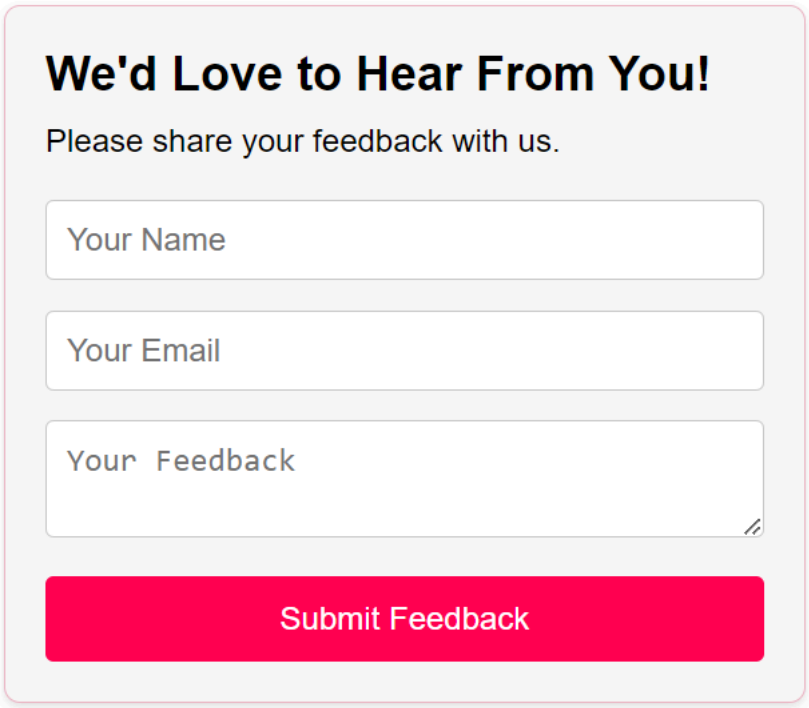
步骤 3：创建基本模板

1. 在你 React 项目的 src 文件夹中导航到 **Components** 文件夹。
2. 导航到位于 React 项目 src 文件夹中的 **FeedbackForm.jsx** 组件。在这个组件的 `return` 中，你会看到一个 `<nav>` 标签和一个 class 名为 **feedback-form** 的 `<form>` 标签，其中包含一个子 `<h2>` 标签和一个 `<p>` 标签。
3. 为了创建反馈表单，你需要包含三个输入字段并创建一个按钮来处理用户的以下信息。
    - 第一个输入框用于用户名

- 第二个输入框用于用户电子邮件 ID
- 第三个输入框用于用户反馈

```
<input
          type="text"
          name="name"
          placeholder="你的名字"
        />
        <input
          type="email"
          name="email"
          placeholder="你的电子邮件"
        />
        <textarea
          name="feedback"
          placeholder="你的反馈"
        ></textarea>
        <button type="submit">提交反馈</button>
```

4. 要查看最新的输出，请在终端中执行 `ctrl+c` 并重新运行应用程序。你将在截图中看到输出。



注意：如果应用程序已经在运行，您只需刷新应用程序页面即可

步骤 4：初始化状态以处理表单数据

1. 在组件顶部集成 React 的 useState 钩子以管理表单数据状态。使用 useState 钩子为表单数据的详细信息创建状态变量，例如姓名、电子邮件和反馈。

```
import React, { useState } from 'react';
```

2. 使用空字符串初始化详细信息。您需要使用 **useState** 钩子初始化一个对象，包括姓名、电子邮件和反馈表单详细信息。

```
const [formData, setFormData] = useState({
    name: '',
    email: '',
    feedback: ''
  });
```

注意：在此组件的 `return` 之前包含上述代码。

步骤 5：实现变更处理程序

1. 定义一个 **handleChange** 函数，以便在用户输入信息时更新表单数据状态。在初始化变量后，使用 **useState** 钩子创建此函数。
2. 在 **handleChange** 函数内部，从事件对象中提取输入字段的名称和值。
3. 使用 **setFormData** 函数和扩展运算符更新表单数据状态，将新值与现有表单数据合并。

```
 const handleChange = (event) => {
  const { name, value } = event.target;
  setFormData({
    ...formData,
    [name]: value
  });
```

```
};
```

- `const handleChange = (event) => { ... }`：这一行声明了一个名为 handleChange 的常量，并将其赋值为一个箭头函数。该函数接受一个事件参数，表示用户与输入元素交互时触发的事件。
- `const { name, value } = event.target;`：这一行从事件对象的目标属性中提取名称和值属性。目标属性指的是触发事件的 DOM 元素，在这种情况下，是一个输入字段。名称属性对应于输入字段的 'name' 属性，而值属性表示用户当前在输入字段中输入的 'value'。
- `setFormData({ ...formData, [name]: value });`：**setFormData** 函数是 React 的 useState 钩子提供的状态更新函数，用于更新状态变量 formData。它展开现有的 formData 对象，然后用新值更新由 'name' 变量指定的属性。这种模式确保状态以不可变的方式更新，意味着创建一个包含更新属性的新对象，而不是直接修改现有状态。

第6步：集成表单状态和onchange事件

1. 在JSX代码中，将表单输入字段名称、电子邮件和反馈绑定到各自的状态变量，使用`formData.name`、`formData.email`和`formData.feedback`。
2. 使用 **value** 属性设置输入字段的值，并使用 **onChange** 属性在输入值更改时调用 **handleChange** 函数。

```
<input
          type="text"
          name="name"
          placeholder="您的姓名"
          value={formData.name}
          onChange={handleChange}
        />
        <input
          type="email"
          name="email"
          placeholder="您的电子邮件"
          value={formData.email}
          onChange={handleChange}
        />
        <textarea
          name="feedback"
          placeholder="您的反馈"
          value={formData.feedback}
          onChange={handleChange}
        ></textarea>
```

注意：在您之前的输入字段和文本区域中包含value和onchange属性。

步骤 7：处理表单提交

1. 通过定义一个 **handleSubmit** 函数来实现表单提交功能。该函数应接受一个事件参数，并阻止默认的表单提交。
2. 然后，创建一个名为 **confirmationMessage** 的变量以捕获用户详情。
3. 接下来，创建另一个变量 **isConfirmed**，用于确认用户的详情是否正确。
4. 如果确认无误，则将表单数据记录到控制台，使用警告框向用户显示 **感谢您** 的消息，并通过重置表单数据状态来清空表单字段。

```
const handleSubmit = (event) => {
    event.preventDefault();
    const confirmationMessage = `
      Name: ${formData.name}
      Email: ${formData.email}
      Feedback: ${formData.feedback}
    `;
    const isConfirmed = window.confirm(`请确认您的信息：\n\n${confirmationMessage}`);
    if (isConfirmed) {
      console.log('提交反馈:', formData);
      setFormData({
        name: '',
        email: '',
        feedback: ''
      });
      alert('感谢您宝贵的反馈！');
    }
  };
```

- 这段代码包含：
  - `const confirmationMessage = ...`: 此模板使用 `formData` 对象中的数据构建确认消息。它包括用户输入的姓名、电子邮件和反馈字段。
  - `const isConfirmed = window.confirm(...);`: 此行使用 `window.confirm()` 显示确认对话框，将 `confirmationMessage` 呈现给用户。如果用户在对话框中确认了信息，`isConfirmed` 被设置为 true；否则，设置为 false。
  - `if (isConfirmed) { ... }`: 此条件语句检查用户是否通过点击确认对话框中的“确定”来确认了他们的详细信息。
  - `setFormData({ ... });`: 调用 `setFormData` 函数将 `formData` 状态重置为空值，在提交后清空表单字段。
  - `alert('感谢您宝贵的反馈！');`: 在重置表单后，向用户显示一个警告，感谢他们的反馈。

注意：将此代码包含在 FeedbackForm 组件的 `return` 之前。

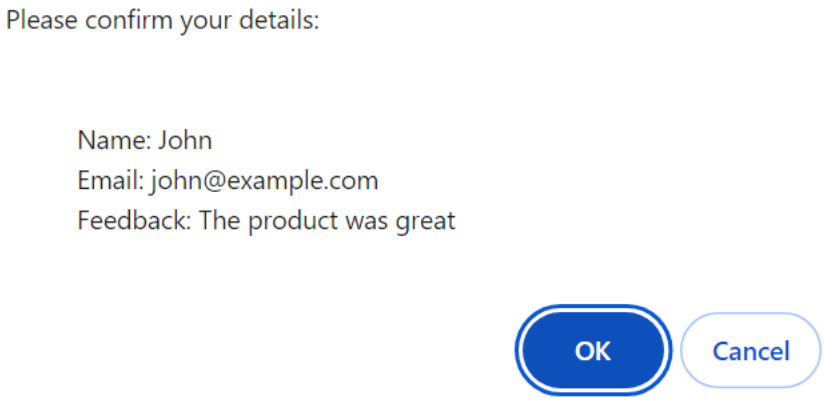
步骤 8：实现 onSubmit 事件处理器

1. 要提交详细信息，请在 `<form>` 标签中包含 `onSubmit` 事件处理器。

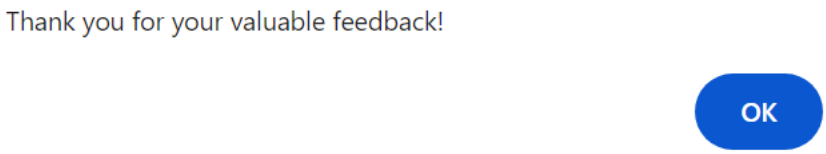
```
<form onSubmit={handleSubmit} className="feedback-form">
```

步骤 9：检查输出

1. 现在，你需要再次运行应用程序并填写详细信息。然后点击 **提交反馈**。将会出现一个确认框，显示填写的详细信息。如果应用程序已在浏览器中运行，只需刷新页面即可。



2. 如果你点击 **确定**，将会显示另一个带有感谢信息的提示框。



3. 如果你在确认框中点击 **取消**，填写的详细信息的表单将会重新出现。

恭喜！你已经创建了一个反馈表单应用程序！

4. ▼ 点击这里查看 FeedbackForm.jsx 的完整代码解决方案

```
import React, { useState } from 'react';
import './FeedbackForm.css'; // 导入 CSS 进行样式设置
const FeedbackForm = () => {
    const [formData, setFormData] = useState({
        name: '',
        email: '',
        feedback: ''
      });
      const handleChange = (event) => {
        const { name, value } = event.target;
        setFormData({
          ...formData,
          [name]: value
        });
      };
      const handleSubmit = (event) => {
        event.preventDefault();
        const confirmationMessage = `
          姓名: ${formData.name}
          邮箱: ${formData.email}
          反馈: ${formData.feedback}
        `;
        const isConfirmed = window.confirm(`请确认您的详细信息:\n\n${confirmationMessage}`);
        if (isConfirmed) {
          console.log('提交反馈:', formData);
          setFormData({
            name: '',
            email: '',
            feedback: ''
          });
          alert('感谢您宝贵的反馈！');
        }
      };
  return (
    <>
    <nav>
    告诉我们您的想法
    </nav>
    <form onSubmit={handleSubmit} className="feedback-form">
        <h2>我们很想听到您的声音！</h2>
        <p>请与我们分享您的反馈。</p>
        <input
          type="text"
          name="name"
```

```
            placeholder="您的姓名"
            value={formData.name}
            onChange={handleChange}
          />
          <input
            type="email"
            name="email"
            placeholder="您的邮箱"
            value={formData.email}
            onChange={handleChange}
          />
          <textarea
            name="feedback"
            placeholder="您的反馈"
            value={formData.feedback}
            onChange={handleChange}
          ></textarea>
          <button type="submit">提交反馈</button>
        </form>
      </>
    );
};
export default FeedbackForm;
```

练习题

1. 在本练习中，您将创建一个评分系统，使用单选按钮让用户提供评分。
2. 首先，您需要在 `formData` **useState** 钩子中创建一个名为 `rating` 的变量，并将其初始化为空字符串。

   提示：将此变量作为键值对包含在 `formData` 中。

▼ 点击这里查看解决方案

```
const [formData, setFormData] = useState({
name: '',
email: '',
feedback: '',
rating: '' // 新的评分状态
    });
```

3. 为此，您需要使用类型为单选按钮的输入框。创建一个包含 **五个** 单选按钮的评分系统，每个单选按钮代表一个从 1 到 5 的评分值。每个单选按钮应显示其对应的值：第一个单选按钮显示“1”，第二个显示“2”，依此类推，直到第五个单选按钮显示“5”。确保类型为单选按钮的输入框具有 `onChange` 事件处理程序，以使用 `handleChange` 函数更新 formData 状态，以反映用户所做的新选择。

   提示：使用类型为 `type='radio'` 的输入框。使用 `onChange={handleChange}` 调用事件处理函数。

▼ 点击这里查看解决方案

```
<div style={{display:'flex',gap:'10px',flexDirection:'column'}}>
                  <span>请给我们评分：</span>
                  <p><input
                      type="radio"
                      name="rating"
                      value="1"
                      onChange={handleChange}
                  /> 1</p>
                 <p>  <input
                      type="radio"
                      name="rating"
                      value="2"
                      onChange={handleChange}
                  /> 2</p>
                 <p>  <input
                      type="radio"
                      name="rating"
                      value="3"
                      onChange={handleChange}
                  /> 3</p>
                  <p> <input
                      type="radio"
                      name="rating"
                      value="4"
                      onChange={handleChange}
                  /> 4</p>
                  <p><input
                      type="radio"
                      name="rating"
                      value="5"
                      onChange={handleChange}
                  /> 5</p>
              </div>
```

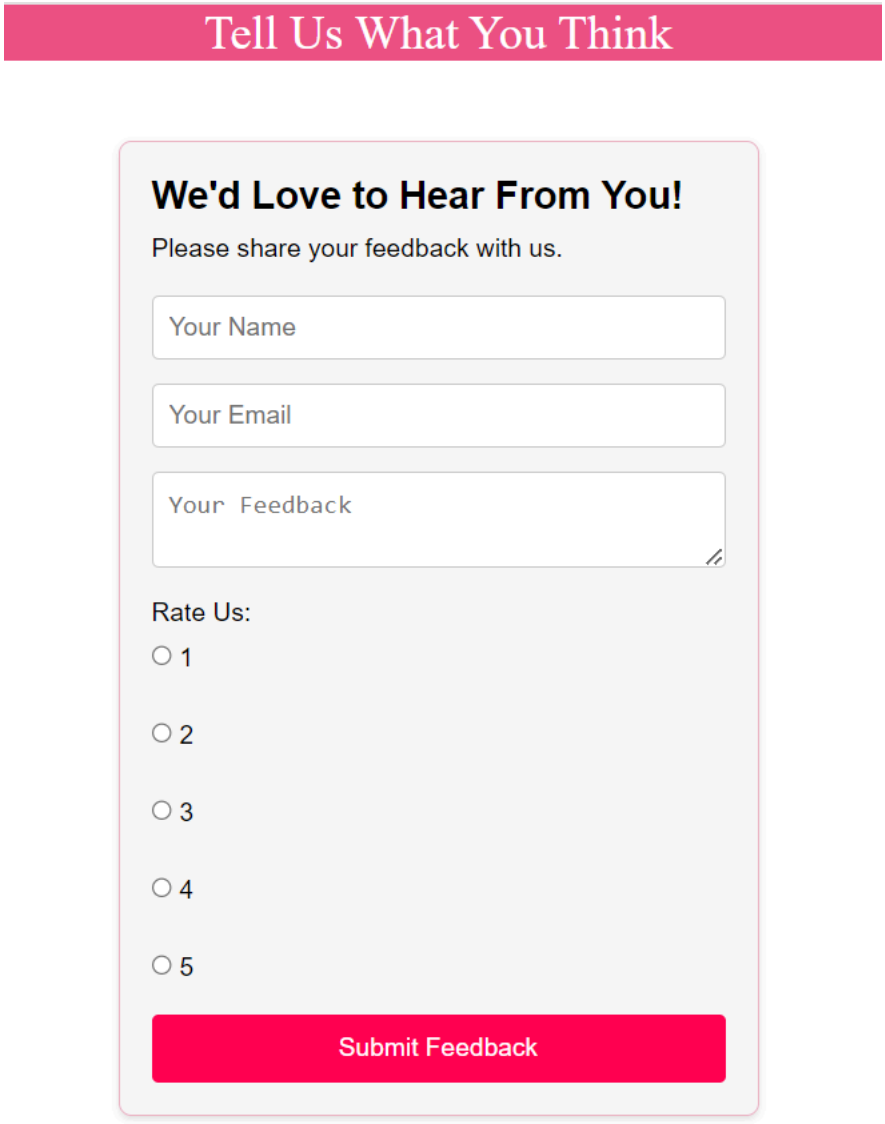
4. `handleChange` 函数将保持不变，因为您使用 `...formData` 扩展运算符获取所有信息。
5. 现在您需要在 `handleSubmit` 函数中加入评分细节。为此，您需要从 `formData` 变量中获取用户选择的评分值，并将其包含在确认消息中，以便与其他细节一起显示。
   提示：将 formData.rating 值放入一个变量中，以访问和显示用户在单选按钮字段中的输入。

   ▼ 点击这里查看解决方案

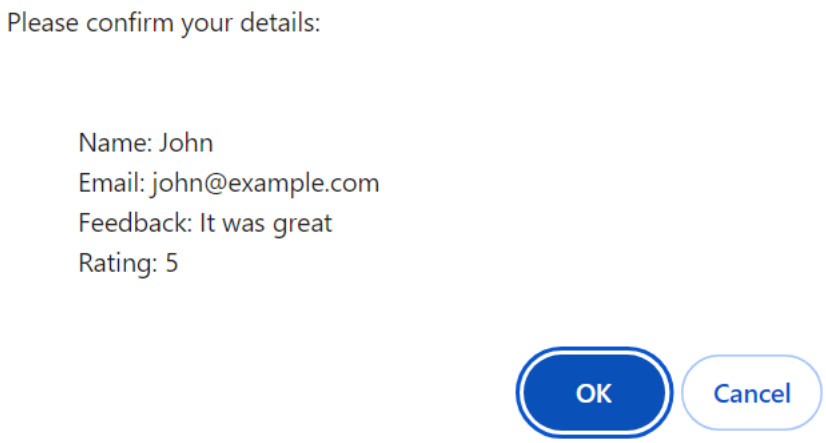
```
const handleSubmit = (event) => {
event.preventDefault();
const confirmationMessage = `
```

```
            姓名: ${formData.name}
            邮箱: ${formData.email}
            反馈: ${formData.feedback}
            评分: ${formData.rating}
        `;
        const isConfirmed = window.confirm(`请确认您的详细信息:\n\n${confirmationMessage}`);
        if (isConfirmed) {
            console.log('提交反馈:', formData);
            setFormData({
                name: '',
                email: '',
                feedback: '',
                rating: '' // 提交后重置评分
            });
            alert('感谢您宝贵的反馈! ');
        }
    };
```

6. 现在通过重新运行应用程序检查输出。输出将与给定的截图相符。



7. 再次填写整个表单并检查警报消息。它将显示评分以及其他细节。



▶ 点击这里查看完整代码解决方案

结论

- 您在给定的 ‘FeedbackForm’ 组件中实现了一个收集用户反馈的表单。该组件利用 React 的 useState 钩子来管理表单数据状态，包括用户的姓名、电子邮件和反馈。
- ‘handleChange’ 函数在用户输入信息到表单字段时更新表单数据状态。提交表单时，’handleSubmit’ 函数阻止默认的表单提交行为，显示包含用户详细信息的确认对话框，清空表单字段，并在确认后显示感谢信息。
- 表单包括用户姓名和电子邮件的输入字段，以及一个用于提供反馈的文本区域。

Author(s)

Richa Arora

© IBM Corporation. All rights reserved.