Task 7: Beginner-Level API Calls in React for a Profile Website

Objective:

Learn to make basic API calls in a React application to fetch and display profile data. This task includes setting up Axios for making API calls, handling responses and errors, and rendering the fetched data in a React component.

Pre-requisites:

- Basic understanding of HTML, CSS, and JavaScript
- Familiarity with a code editor like Visual Studio Code
- Basic knowledge of React

Concepts Covered:

- Setting Up Axios and Making a Basic API Call
- Displaying Fetched Data in a Component
- Basic Error Handling in API Requests

Concepts:

1. **Installing Axios and Understanding API Calls:**
   Install Axios and explore its documentation briefly to understand how to make API calls.

```
npm install axios
```

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;
    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

2. **Fetching Data from a Mock API:**

   Use Axios to make a simple GET request to fetch data from a mock API (like JSONPlaceholder or a similar service) for profile information.

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

3. **Displaying API Data in a Component:**

   Create a `ProfileList` component and display the fetched profile data in a list format.

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

4. **Simple Error Handling in API Requests:**

Implement basic error handling for your Axios request to manage any failed API calls gracefully.

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

Setup:

1. **Install Visual Studio Code (VS Code):**

   Download and install VS Code from Visual Studio Code.

2. **React Developer Tools:**

   Install the React Developer Tools browser extension for Chrome or Firefox.

3. **Git:**

   Install Git for version control. You can download it from Git.

4. **Node.js and npm:**

   Ensure Node.js and npm are installed on your machine. You can download and install them from Node.js.

5. **Setting Up the React Project:**
   - Open your terminal.
   - Navigate to the directory where you want to create your project.
   - Run the following command to create a new React project:

   ```
   npx create-react-app profile-website
   ```

6. **Installing Additional Dependencies:**
   - Navigate to the project directory:

   ```
   cd profile-website
   ```

   - Install Axios:

   ```
   npm install axios
   ```

Tasks:

Part 1: Setting Up Axios and Making a Basic API Call (30 minutes)

1. **Installing Axios and Understanding API Calls (15 minutes):**
   - Install Axios and explore its documentation briefly.
   - Example:

   ```
   npm install axios
   ```

2. **Fetching Data from a Mock API (15 minutes):**
   - Use Axios to make a simple GET request to fetch data from a mock API (like JSONPlaceholder or a similar service) for profile information.
   - Example:

   ```
   // src/components/ProfileList.js
   import React, { useEffect, useState } from 'react';
   import axios from 'axios';

   const ProfileList = () => {
       const [profiles, setProfiles] = useState([]);
       const [loading, setLoading] = useState(true);
       const [error, setError] = useState(null);
   ```

```
    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</
li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

- Goal: Learn to perform a basic HTTP GET request and handle the response to retrieve data.

Part 2: Displaying Fetched Data and Basic Error Handling (30 minutes)

1. **Displaying API Data in a Component (15 minutes):**
    - Create a `ProfileList` component and display the fetched profile data in a list format.
    - Example:

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        axios.get('https://jsonplaceholder
                  .typicode.com/users')
.then(response => {
setProfiles(response.data);
setLoading(false);
})
.catch(error => {
setError(error.message);
setLoading(false);
});
}, []);
```

```
    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```
```

- Goal: Understand how to render data fetched from an API onto the UI.

2. **Simple Error Handling in API Requests (15 minutes):**
    - Implement basic error handling for your Axios request to manage any failed API calls gracefully.
    - Example:

```
// src/components/ProfileList.js
import React, { useEffect, useState } from 'react';
import axios from 'axios';

const ProfileList = () => {
    const [profiles, setProfiles] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);
```
```

```
    useEffect(() => {
        axios.get('https://jsonplaceholder.typicode.com/users')
            .then(response => {
                setProfiles(response.data);
                setLoading(false);
            })
            .catch(error => {
                setError(error.message);
                setLoading(false);
            });
    }, []);

    if (loading) return <p>Loading...</p>;
    if (error) return <p>Error: {error}</p>;

    return (
        <div>
            <h2>Profile List</h2>
            <ul>
                {profiles.map(profile => (
                    <li key={profile.id}>{profile.name} - {profile.email}</
li>
                ))}
            </ul>
        </div>
    );
};

export default ProfileList;
```

- Goal: Learn to handle errors that may occur during API requests and display a user-friendly message.

Instructions:

1. Write the required code in the appropriate files in the `src/components` directory.
2. Open the terminal and navigate to the project directory.
3. Run the React project using:

```
npm start
```

4. Open the project in a web browser to ensure the code displays correctly.
5. Use the browser's developer tools and React Developer Tools to debug and inspect the elements.

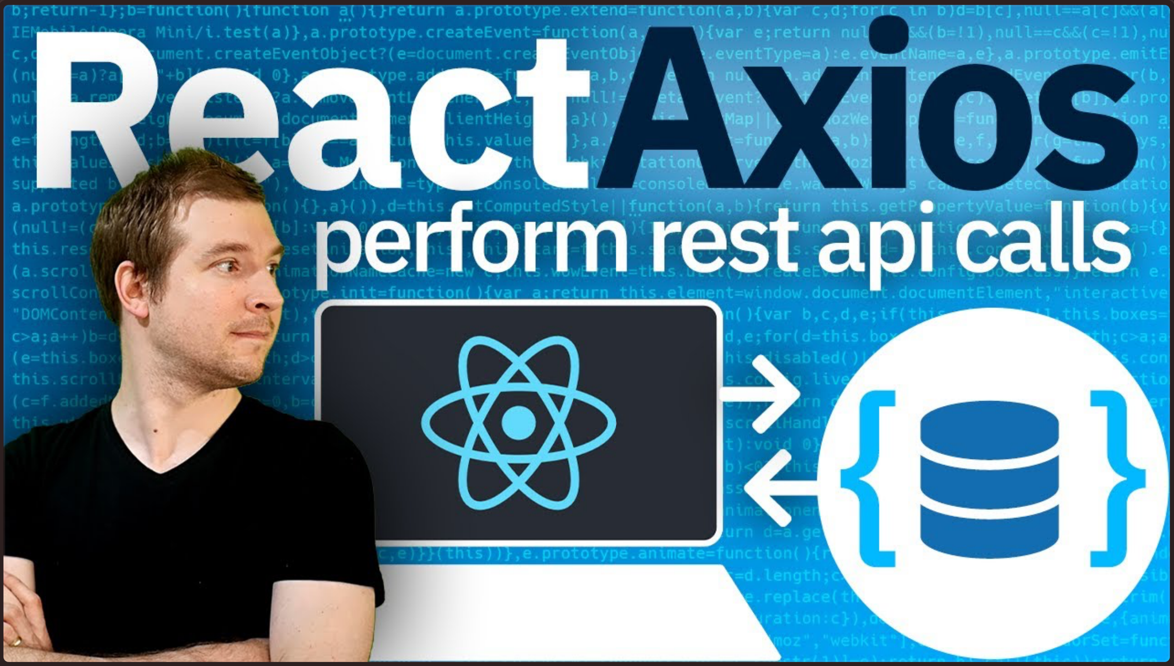
Resources:

- Getting Started | Axios Docs
  Axios is a promise-based HTTP Client for node.js and the browser. It is isomorphic (= it can run in the...
  https://axios-http.com/docs/intro
- Getting Started – React
  A JavaScript library for building user interfaces
  https://reactjs.org/docs/getting-started.html
- Create React App
  Set up a modern web app by running one command.
  https://create-react-app.dev/
- Get started with Visual Studio Code
  Documentation for Visual Studio Code
  Find out how to set-up and get the most from Visual Studio Code. Optimized for building and d...
  https://code.visualstudio.com/docs

Videos:

-

  Traversy Media 42:20min 365,019 Views 8,284 Likes

- 

Adrian Twarog 10:38 min 289,675 Views 3,936 Likes

GitHub Instructions:

1. **Open in Visual Studio Code:**
   After clicking on the "Open in Visual Studio Code" button from the GitHub Classroom confirmation page, VSCode will open the repository directly. If prompted, select "Open" or "Allow" to open the repository in VSCode.
2. **Open the Terminal in VSCode:**
   In VSCode, open a terminal by selecting Terminal > New Terminal from the top menu.
3. **Complete the Task:**
   In VSCode, write your solution in the appropriate files in the `src/components` directory.
4. **Run and Test Your Code:**
   Open your terminal, navigate to your project directory, and run:

```
npm start
```

5. **Commit Your Changes:**
   In the VSCode terminal, add your changes to git:

```
git add src/components/*
```

   Commit your changes with a meaningful message:

```
git commit -m "Completed task 27"
```

6. **Push Your Changes to Your Repository:**

   Push your changes to your forked repository:

   ```
   git push origin main
   ```

7. **Create a Pull Request:**

   Go to your repository on GitHub.

   Click on the "Pull Requests" tab.

   Click the "New Pull Request" button.

   Ensure the base repository is the original template repository and the base branch is `main` .

   Ensure the head repository is your forked repository and the compare branch is `main` .

   Click "Create Pull Request".

   Add a title and description for your pull request and submit it.

Summary of Commands:

```
# Open in Visual Studio Code

# Open terminal in VSCode

# Complete the task by editing src/components/*

# Navigate to the project directory
cd profile-website

# Run your code
npm start

# Add, commit, and push your changes
git add src/components/*
git commit -m "Completed task 7"
git push origin main

# Create a pull request on GitHub
```