**Assignment 2: Asynchronous Operations in React**

**Handle asynchronous operations in React (e.g., API calls).**

**Use async/await or promises for better code readability.**

Handling asynchronous operations in React, especially API calls, is a common task. You can use either async/await or promises to handle asynchronous operations. Below, I'll provide examples for both approaches.

Using async/await:

| **// src/App.js**  **import React, { useState, useEffect } from 'react';**  **import axios from 'axios';**  **function App() {**  **const [data, setData] = useState([]);**  **const [loading, setLoading] = useState(true);**  **useEffect(() => {**  **const fetchData = async () => {**  **try {**  **const response = await axios.get('http://localhost:3001/api/data');**  **setData(response.data);**  **setLoading(false);**  **} catch (error) {**  **console.error('Error fetching data:', error);**  **setLoading(false);**  **}**  **};**  **fetchData();**  **}, []); // Empty dependency array to run the effect only once**  **return (**  **<div>**  **<h1>Data from Express.js API</h1>**  **{loading ? (**  **<p>Loading...</p>**  **) : (**  **<ul>**  **{data.map((item) => (**  **<li key={item.id}>{item.name}</li>**  **))}**  **</ul>**  **)}**  **</div>**  **);**  **}**  **export default App;** |
| --- |

**In this example, the fetchData function is declared as async, allowing the use of await with the asynchronous axios.get call. This makes the code more readable and similar to synchronous code.**

**Using Promises:**

| **// src/App.js**  **import React, { useState, useEffect } from 'react';**  **import axios from 'axios';**  **function App() {**  **const [data, setData] = useState([]);**  **const [loading, setLoading] = useState(true);**  **useEffect(() => {**  **const fetchData = () => {**  **axios**  **.get('http://localhost:3001/api/data')**  **.then((response) => {**  **setData(response.data);**  **setLoading(false);**  **})**  **.catch((error) => {**  **console.error('Error fetching data:', error);**  **setLoading(false);**  **});**  **};**  **fetchData();**  **}, []); // Empty dependency array to run the effect only once**  **return (**  **<div>**  **<h1>Data from Express.js API</h1>**  **{loading ? (**  **<p>Loading...</p>**  **) : (**  **<ul>**  **{data.map((item) => (**  **<li key={item.id}>{item.name}</li>**  **))}**  **</ul>**  **)}**  **</div>**  **);**  **}**  **export default App;** |
| --- |

**In this example, the axios.get call returns a Promise, and the .then and .catch methods are used to handle the successful and failed outcomes, respectively.**

**Both approaches are valid, and the choice between them often comes down to personal preference or team conventions. Using async/await can make your code more readable and maintainable, especially when dealing with multiple asynchronous operations. However, using promises is still a common and valid pattern in React applications.**