

**Lab Manual- React Project 01**

**Prepared for**:

**Date:** 18th June 2024

**Prepared by:**

Document Name: Lab Manual **Document Number** SDLab333

**Contributor:**

Contents

[1. What are Axios 3](#_Toc172114281)

[2. Create New React Project 3](#_Toc172114282)

[3. Adding Axios to the Project 4](#_Toc172114283)

[4. Making a GET Request 4](#_Toc172114284)

[1. Explanation: 6](#_Toc172114285)

[5. Step 3 — Making a POST Request 7](#_Toc172114286)

# What are Axios

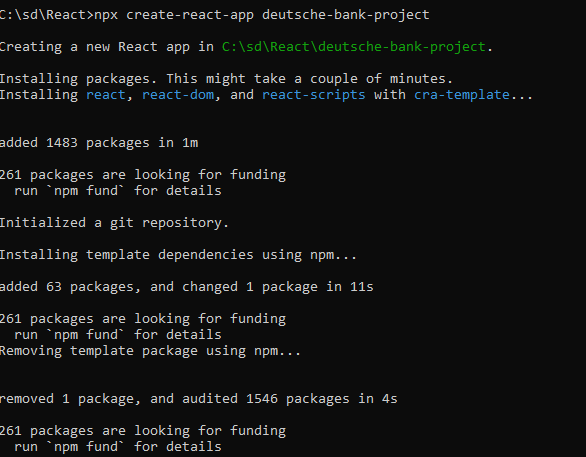
1. **Axios Basics**: Axios is a JavaScript library used for making HTTP requests (like fetching data) from web servers or APIs in web applications. It's similar to the Fetch API but provides more features and is promise-based.
2. **Promise-based**: Axios uses promises, which allow you to handle asynchronous operations in a more readable way using async and await keywords. This makes it easier to write and manage code that deals with asynchronous tasks like fetching data.
3. **Interception and Cancellation**: Axios allows you to intercept requests and responses, which means you can modify them or handle errors before they reach your application. You can also cancel requests, which is useful for scenarios where you want to stop a request that is no longer needed.
4. **CSRF Protection**: Axios includes built-in protection against Cross-Site Request Forgery (CSRF) attacks on the client-side. This helps ensure that requests made from your application are secure and not exploited by malicious actors.

In summary, Axios simplifies making HTTP requests in JavaScript applications, supports modern async programming with promises, provides advanced features like request interception and cancellation, and enhances security with built-in CSRF protection. It's widely used in web development for its simplicity and robust functionality.

# Create New React Project

* Open Command Prompt in Administrative Mode and Type below command

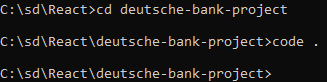
**npx create-react-app deutsche-bank-project**

****

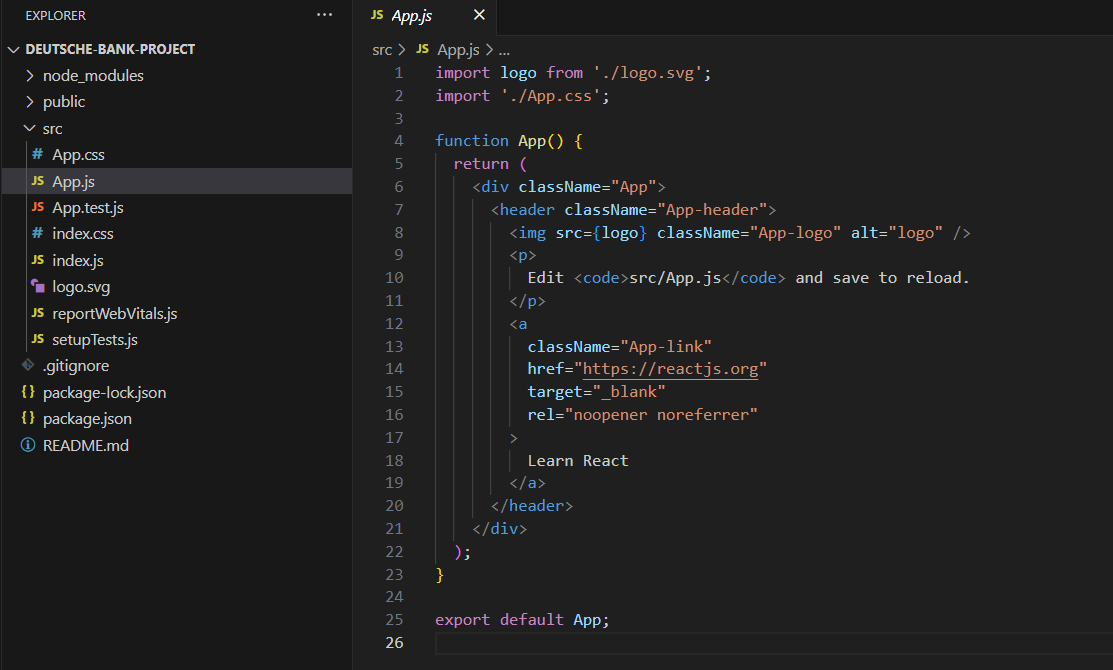
* Go Inside Project Directory and Type **Code .**

**cd deutsche-bank-project**

**code .**

****

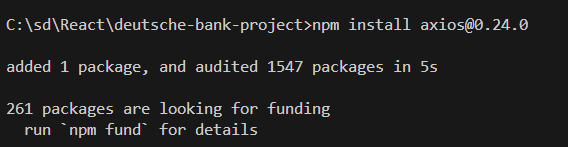
* Now you are inside project



# [Adding Axios to the Project](https://www.digitalocean.com/community/tutorials/react-axios-react#step-1-adding-axios-to-the-project)

run this command to install Axios:

**npm install axios@0.24.0**

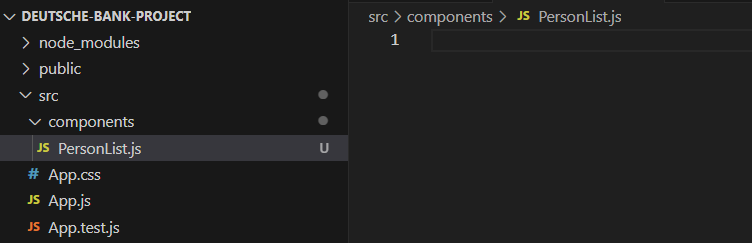
****

# [Making a GET Request](https://www.digitalocean.com/community/tutorials/react-axios-react#step-2-making-a-get-request)

In this example, you create a new component and import Axios into it to send a **GET** request.

* Inside your React project, you will need to create a new component named **PersonList**.

First, create a new **components** subdirectory in the **src** directory:



* Add the following code to the component **PersonList.js**

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

import axios from 'axios';

const PersonList = () => {

  const [persons, setPersons] = useState([]);

  useEffect(() => {

    axios.get(`https://jsonplaceholder.typicode.com/users`)

      .then(res => {

        const personsData = res.data;

        setPersons(personsData);

      });

  }, []); // Empty dependency array ensures useEffect runs once on component mount

  return (

    <ul>

      {

        persons.map(person =>

          <li key={person.id}>{person.name}</li>

        )

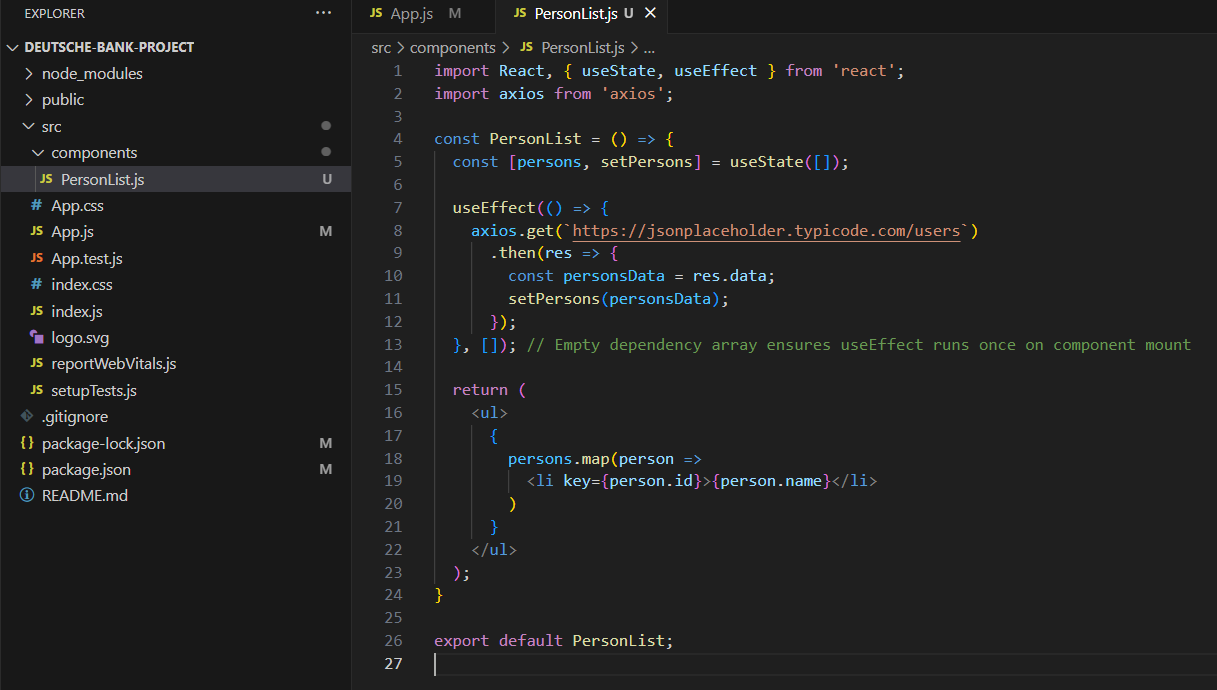
      }

    </ul>

  );

}

export default PersonList;



### Explanation:

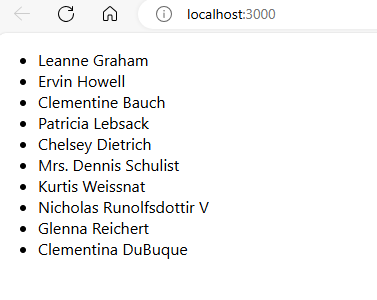
1. **useState**: useState hook is used to initialize persons state as an empty array ([]). It replaces this.state and this.setState in class components.
2. **useEffect**: useEffect hook is used to perform side effects in function components. Here, it replaces componentDidMount lifecycle method. It runs once when the component mounts ([] as dependency array ensures it runs only once).
3. **Axios Request**: Inside useEffect, Axios makes a GET request to fetch data from the specified URL (https://jsonplaceholder.typicode.com/users). When data is fetched successfully (then block), it updates the persons state using setPersons.
4. **Rendering**: In the return statement, persons state is mapped to render a list (<ul>) of person names (<li> elements with person.name). Each list item has a unique key (person.id) to help React efficiently update the UI.

This approach leverages React hooks for managing state and lifecycle in a concise and modern way compared to class components.

* Then run your application:

**npm start**

* View the application in the browser. You will be presented with a list of 10 names from the website [jsonplaceholder.typicode.com/users](https://jsonplaceholder.typicode.com/users)



# [Step 3 — Making a POST Request](https://www.digitalocean.com/community/tutorials/react-axios-react#step-3-making-a-post-request)

In this step, you will use Axios with another HTTP request method called POST.

* Inside your React project, you will need to create a new component named PersonAdd.

Create PersonAdd.js and add the following code to create a form that allows for user input and subsequently POSTs the content to an API:

import React, { useState } from 'react';

import axios from 'axios';

const PersonAdd = () => {

  const [name, setName] = useState('');

  const handleChange = event => {

    setName(event.target.value);

  };

  const handleSubmit = event => {

    event.preventDefault();

    console.log('Submitting form with name:', name); // Check if handleSubmit is called

    const user = {

      name: name

    };

    axios.post(`https://jsonplaceholder.typicode.com/users`, user)

      .then(res => {

        console.log('Post request successful:', res.data); // Check response data

      })

      .catch(error => {

        console.error('Error adding person:', error); // Log any errors

      });

  };

  return (

    <div>

      <form onSubmit={handleSubmit}>

        <label>

          Person Name:

          <input type="text" value={name} onChange={handleChange} />

        </label>

        <button type="submit">Add</button>

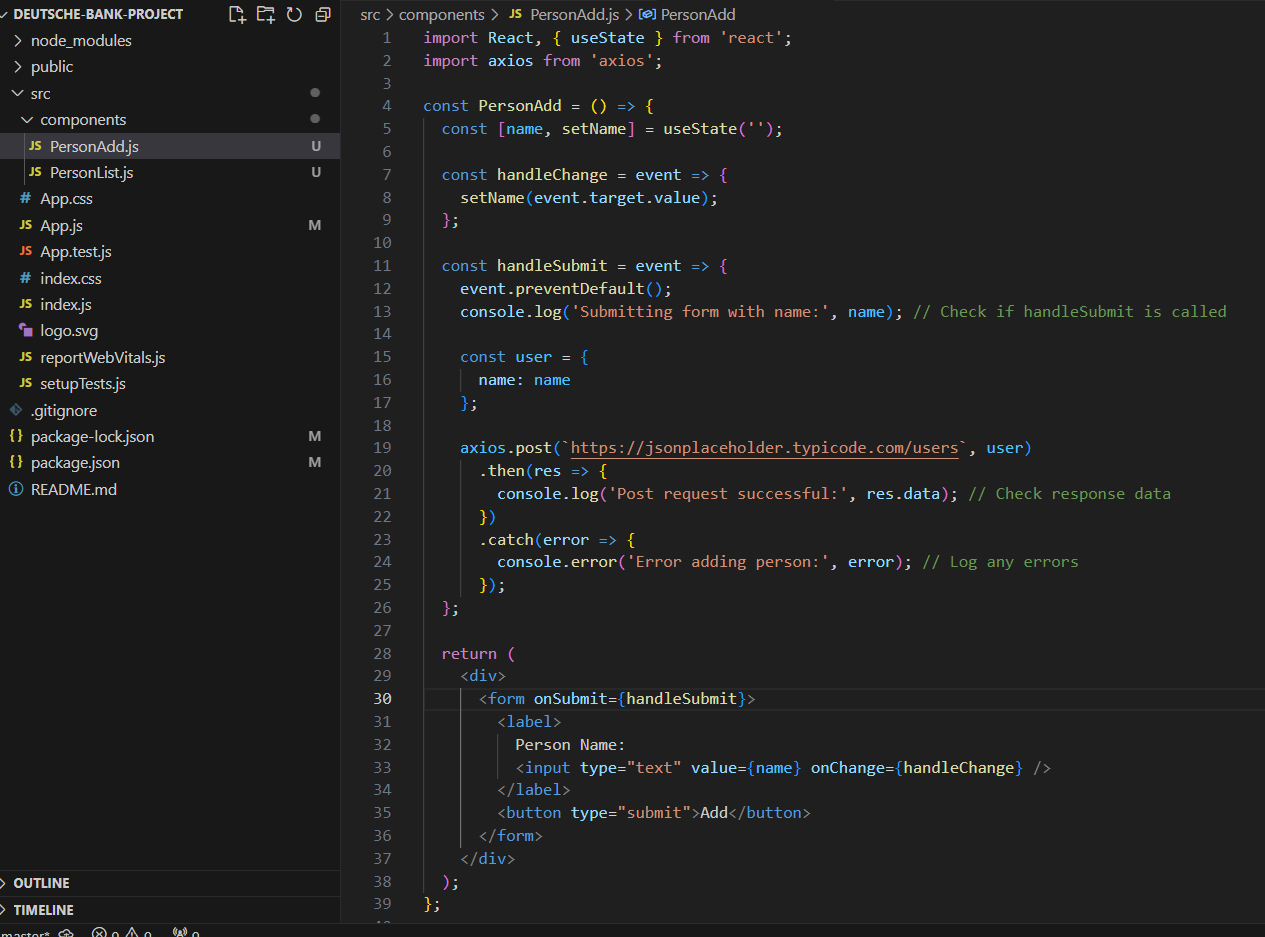
      </form>

    </div>

  );

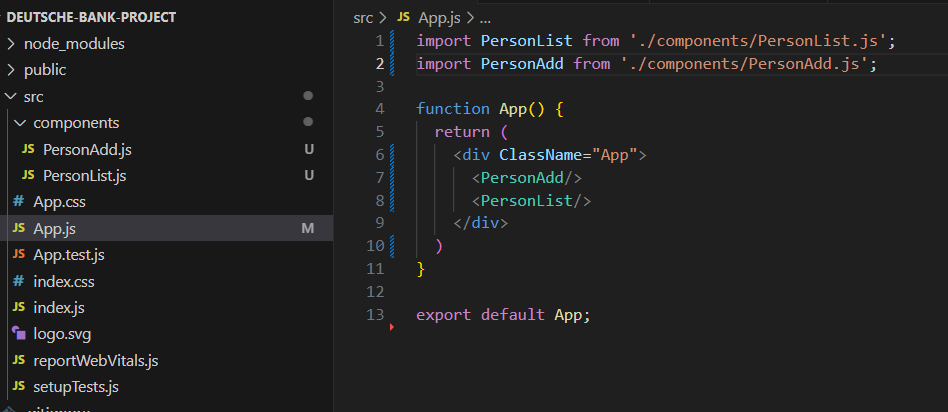
};

export default PersonAdd;



To complete the POST request, you first capture the user input. Then you add the input along with the POST request, which will give you a response

* Add this component to your **app.js:**



* Type something into the input field and check if the console logs show the correct name value when you submit the form.

