**Services**

**What is a service in React?**

In React, a **service** typically refers to a **module** or a set of functions that are responsible for handling data-related operations.

These operations often involve making HTTP requests to a backend server, interacting with APIs, or performing any other data-fetching and data-manipulating tasks.

By separating these concerns into a service, you promote modularity, reusability, and cleaner code.

**Key Benefits of Using Services in React:**

1. **Separation of Concerns:** Services help keep your components clean and focused on presentation logic, while data-fetching and other business logic are handled separately.
2. **Reusability:** Functions within services can be reused across multiple components, avoiding duplication of code.
3. **Maintainability**: It becomes easier to manage and update your data-fetching logic without affecting the component logic.

**What is axios?**

**Axios** is a popular, promise-based HTTP client for JavaScript that is used to make HTTP requests from the browser and Node.js.

It simplifies the process of making HTTP requests, such as **GET**, **POST**, **PUT**, **DELETE**, and handling the responses, including error handling.

**Key Features of Axios:**

1. **Promise-based**: Axios uses promises, which makes it easy to work with asynchronous requests and handle responses using **.then()** and **.catch()**.
2. **Support for Request and Response Interception**: You can intercept requests or responses before they are handled, allowing for features like logging, modifying requests, or error handling.
3. **Automatic JSON Data Transformation**: Axios automatically transforms JSON data, making it easy to send and receive JSON objects.
4. **Request and Response Data Transformation**: You can transform the data before sending it to the server or after receiving it from the server.
5. **Automatic CSRF Protection**: Axios automatically handles Cross-Site Request Forgery (CSRF) protection by sending the CSRF token as a header.
6. **Node.js Support**: Axios works in both browser environments and Node.js, making it a versatile choice for both client-side and server-side HTTP requests.
7. **Cancellation**: You can cancel requests using the **CancelToken** feature, which is useful for preventing unnecessary network activity.
8. **Wide Browser Compatibility**: Axios works in all modern browsers and is compatible with older ones as well.

**Installation**

To use Axios in a project, you first need to install it via npm or yarn:

**npm install axios@latest**

**What is FetchAPI?**

The **Fetch API** is a modern interface that allows you to make network requests similar to XMLHttpRequest (XHR).

It is more powerful and flexible, providing a better way to make asynchronous requests using promises.

The Fetch API is built into most modern browsers and can be used to request resources such as data, images, or other assets from a server.

Key Features of Fetch API:

1. **Promise**-**based**: Fetch uses promises, which makes it easier to work with asynchronous requests and handle responses using .then() and .catch().
2. **Simplified** **API**: Fetch provides a more straightforward and clean API compared to the older **XMLHttpRequest**.
3. **Streamed** **Response**: Fetch allows you to work with response streams, enabling you to handle large files or process data chunks as they are received.
4. **Better** **Abstraction**: Fetch abstracts away much of the complexity involved in making HTTP requests, like handling different types of responses, setting headers, etc.
5. **More** **Consistent** **Handling** **of** **Errors**: Fetch does not reject HTTP errors (like 404 or 500); instead, it resolves the promise and leaves it up to the user to handle them. This allows for more consistent error handling

**Making HTTP requests in React**

Making HTTP requests in React typically involves using JavaScript libraries like Axios, the Fetch API, or other similar tools.

Below, I will explain how to make HTTP requests using both Axios and the Fetch API.

**Using Axios**

**Installation**

First, you need to install Axios. You can do this using npm

**npm install axios@latest**

**What is useEffect() in React?**

**useEffect** is a hook in React that allows you to perform side effects in function components.

Side effects can include data fetching, subscriptions, or manually changing the DOM in React components.

**useEffect** serves as a combination of **componentDidMount**, **componentDidUpdate**, and **componentWillUnmount** lifecycle methods in class components.

**Basic Syntax**

import React, { useEffect } from 'react';

useEffect(() => {

// Code to run on component mount and update

return () => {

// Code to run on component unmount

};

}, [dependencies]);

**Explanation of Parameters**

* **Effect Function**: The first argument to **useEffect** is a function that contains the code you want to run. This function can optionally return a cleanup function.
* **Dependency Array**: The second argument is an optional array of dependencies. The effect function will only run when one of the dependencies has changed.

**How useEffect Works**

1. **On Mount**: If you provide an empty dependency array (**[]**), the effect function will run only once after the initial render.
2. **On Update**: If you provide a dependency array with values, the effect function will run after every render where any of the dependencies have changed.
3. **Cleanup**: The cleanup function runs before the component is unmounted or before the effect is re-executed due to changes in dependencies.

**Example 01**

Here's an example of how to use **Axios** to make a **GET** request to fetch data and a **POST** request to submit data in a React component.

DataFetcherAxios.js

// Import the necessary libraries

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

import axios from 'axios';

import './DataFetcherAxios.css';

const DataFetcherAxios = () => {

  const [data, setData] = useState([]);

  const [postData, setPostData] = useState({

    title: '',

    userId: 1,

  });

  // Function to fetch data from an API

  useEffect(() => {

    axios.get('https://jsonplaceholder.typicode.com/posts')

      .then(response => {

        setData(response.data);

      })

      .catch(error => {

        console.error('Error fetching data:', error);

      });

  }, []);

  // Function to submit data to an API

  const handleSubmit = (e) => {

    e.preventDefault();

    axios.post('https://jsonplaceholder.typicode.com/posts', postData)

      .then(response => {

        console.log('Data submitted:', response.data);

        // Append the new post to the existing posts

        setData([response.data, ...data]);

      })

      .catch(error => {

        console.error('Error submitting data:', error);

      });

  };

  // Function to handle input changes

  const handleChange = (e) => {

    const { name, value } = e.target;

    setPostData({ ...postData, [name]: value });

  };

  return (

    <div className="form-container">

      <h2>Submit Data:</h2>

      <form onSubmit={handleSubmit}>

        <div>

          <label>Title:</label>

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

        </div>

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

      </form>

      <h2>Data from API:</h2>

      <ul className="data-list">

        {data.map(item => (

          <li key={item.id}>{item.title}</li>

        ))}

      </ul>

    </div>

  );

};

export default DataFetcherAxios;

DataFetcherAxios.css

/\* DataFetcher.css \*/

.form-container {

    max-width: 600px;

    margin: 0 auto;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

  }

  .form-container h2 {

    margin-top: 0;

    margin-bottom: 20px;

  }

  .form-container label {

    display: block;

    margin-bottom: 5px;

  }

  .form-container input[type='text'],

  .form-container textarea {

    width: 100%;

    padding: 8px;

    margin-bottom: 10px;

  }

  .form-container button {

    background-color: #007bff;

    color: #fff;

    border: none;

    padding: 10px 20px;

    cursor: pointer;

  }

  .form-container button:hover {

    background-color: #0056b3;

  }

  .data-list {

    list-style: none;

    padding: 0;

  }

  .data-list li {

    padding: 10px;

    border-bottom: 1px solid #ccc;

  }

App.js

import React from 'react';

import DataFetcherAxios from './DataFetcherAxios';

const App = () => {

  return (

    <div>

      <h1>My React App</h1>

      <DataFetcherAxios />

    </div>

  );

};

export default App;

**Using Fetch API**

The Fetch API is built into modern browsers and doesn't require additional installation.

Here's how to use it:

**Example**

Here's an example of how to use the Fetch API to make a GET request to fetch data and a POST request to submit data in a React component.

**DataFetcherFetch.js**

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

import './DataFetcherFetch.css';

const DataFetcherFetch = () => {

  const [data, setData] = useState([]);

  const [postData, setPostData] = useState({

    title: '',

    userId: 1,

  });

  // Function to fetch data from an API

  useEffect(() => {

    fetch('https://jsonplaceholder.typicode.com/posts')

      .then(response => response.json())

      .then(data => {

        setData(data);

      })

      .catch(error => {

        console.error('Error fetching data:', error);

      });

  }, []);

  // Function to submit data to an API

  const handleSubmit = (e) => {

    e.preventDefault();

    fetch('https://jsonplaceholder.typicode.com/posts', {

      method: 'POST',

      headers: {

        'Content-Type': 'application/json',

      },

      body: JSON.stringify(postData),

    })

      .then(response => response.json())

      .then(newData => {

        console.log('Data submitted:', data);

        // Update the state with the new post

        setData([newData, ...data]);

      })

      .catch(error => {

        console.error('Error submitting data:', error);

      });

  };

  // Function to handle input changes

  const handleChange = (e) => {

    const { name, value } = e.target;

    setPostData({ ...postData, [name]: value });

  };

  return (

    <div className="container">

      <h2>Submit Data:</h2>

      <form onSubmit={handleSubmit} className="form">

        <div className="form-group">

          <label>Title:</label>

          <input

            type="text"

            name="title"

            value={postData.title}

            onChange={handleChange}

            className="form-control"

          />

        </div>

        <button type="submit" className="btn-submit">Submit</button>

      </form>

      <h2>Data from API:</h2>

      <ul className="post-list">

        {data.map(item => (

          <li key={item.id} className="post-item">{item.title}</li>

        ))}

      </ul>

    </div>

  );

};

export default DataFetcherFetch;

**DataFetcherFetch.css**

/\* Container styling \*/

.container {

    width: 80%;

    margin: 0 auto;

    padding: 20px;

    background-color: #f9f9f9;

    border-radius: 8px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  }

  /\* Form styling \*/

  .form {

    margin-bottom: 20px;

  }

  .form-group {

    margin-bottom: 15px;

  }

  .form-group label {

    display: block;

    font-weight: bold;

    margin-bottom: 5px;

  }

  .form-control {

    width: 100%;

    padding: 8px;

    box-sizing: border-box;

    border: 1px solid #ccc;

    border-radius: 4px;

  }

  .btn-submit {

    padding: 10px 20px;

    background-color: #007bff;

    color: #fff;

    border: none;

    border-radius: 4px;

    cursor: pointer;

  }

  .btn-submit:hover {

    background-color: #0056b3;

  }

  /\* Post list styling \*/

  .post-list {

    list-style-type: none;

    padding: 0;

  }

  .post-item {

    padding: 10px;

    border-bottom: 1px solid #ddd;

  }

  .post-item:nth-child(even) {

    background-color: #f1f1f1;

  }

**App.js**

import React from 'react';

import DataFetcherFetch from './DataFetcherFetch';

const App = () => {

  return (

    <div>

      <h1>My React App</h1>

      <DataFetcherFetch />

    </div>

  );

};

export default App;

**Explanation**

1. **State Management:**
   * **useState** is used to manage the state of the data fetched from the API (**data**) and the data to be submitted (**postData**).
2. **Fetching Data (GET Request):**
   * In **useEffect**, a GET request is made to an API endpoint (**https://jsonplaceholder.typicode.com/posts**).
   * The data fetched from the API is then stored in the state variable **data** using **setData**.
3. **Submitting Data (POST Request):**
   * The **handleSubmit** function is called when the form is submitted.
   * This function makes a POST request to an API endpoint with the data stored in **postData**.
   * The response from the API is logged to the console.
4. **Handling Input Changes:**
   * The **handleChange** function updates the **postData** state whenever the input fields are changed.
5. **Rendering the Component:**
   * The component renders a list of items fetched from the API and a form to submit new data.

This structure allows you to handle both fetching and submitting data in a clean and organized way within a React component.

**Example 02**

Let's build a more complex example. In this example, we will create a simple CRUD (Create, Read, Update, Delete) application that interacts with a REST API.

**REST API:**

They allow communication between different applications over the internet.

REST (which stands for **Representational State Transfer**) APIs operate on a stateless client-server architecture, providing a standardized way to create, read, update, and delete resources.

We will use **jsonplaceholder.typicode.com** as our mock API.

The application will have the following features:

1. Display a list of posts.
2. Create a new post.
3. Update an existing post.
4. Delete a post.

Here is how you can set it up:

1. **Create the PostServiceAxios.js file**: This file will contain all the HTTP request functions.

// src/PostServiceAxios.js

import axios from "axios";

const API\_URL = 'https://jsonplaceholder.typicode.com/posts';

export const fetchPosts = () => {

    return axios.get(API\_URL);

};

export const createPost = (postData) => {

    return axios.post(API\_URL, postData);

};

export const updatePost = (id, postData) => {

    return axios.put(`${API\_URL}/${id}`, postData);

}

export const deletePost = (id) => {

    return axios.delete(`${API\_URL}/${id}`);

}

Let's break down the program into its components and explain each part in detail.

**1. Importing Axios**

**import axios from "axios";**

* **axios**: A promise-based HTTP client for the browser and Node.js. It's used to make HTTP requests to interact with APIs.
* **import axios from "axios";**: This line imports the Axios library into your JavaScript file so you can use it to make HTTP requests.

**2. Defining the API URL**

**const API\_URL = 'https://jsonplaceholder.typicode.com/posts';**

* **const API\_URL**: A constant variable holding the base URL of the API endpoint. In this case, it points to the posts endpoint of the JSONPlaceholder API, a free fake online REST API for testing and prototyping.

**3. Fetch Posts Function**

**export const fetchPosts = () => {**

**return axios.get(API\_URL);**

**};**

* **export const fetchPosts = () => { ... };**: Defines and exports a function named **fetchPosts**.
* **axios.get(API\_URL)**: Uses Axios to send a GET request to the **API\_URL** to retrieve all posts.
* **return axios.get(API\_URL);**: Returns the Axios promise. This allows the caller to handle the asynchronous response (success or failure) using **.then()** and **.catch()** methods.

**4. Create Post Function**

**export const createPost = (postData) => {**

**return axios.post(API\_URL, postData);**

**};**

* **export const createPost = (postData) => { ... };**: Defines and exports a function named **createPost**.
* **postData**: A parameter representing the data to be sent in the body of the POST request.
* **axios.post(API\_URL, postData)**: Uses Axios to send a POST request to the **API\_URL**, with **postData** as the payload. This creates a new post.
* **return axios.post(API\_URL, postData);**: Returns the Axios promise. This allows the caller to handle the asynchronous response (success or failure) using **.then()** and **.catch()** methods.

**5. Update Post Function**

**export const updatePost = (id, postData) => {**

**return axios.put(`${API\_URL}/${id}`, postData);**

**};**

* **export const updatePost = (id, postData) => { ... };**: Defines and exports a function named **updatePost**.
* **id**: A parameter representing the ID of the post to be updated.
* **postData**: A parameter representing the new data for the post.
* **axios.put(${API\_URL}/${id}, postData)**: Uses Axios to send a PUT request to the **API\_URL** with the specific **id**, and **postData** as the payload. This updates the existing post.
* **return axios.put(${API\_URL}/${id}, postData);**: Returns the Axios promise. This allows the caller to handle the asynchronous response (success or failure) using **.then()** and **.catch()** methods.

**6. Delete Post Function**

**export const deletePost = (id) => {**

**return axios.delete(`${API\_URL}/${id}`);**

**};**

* **export const deletePost = (id) => { ... };**: Defines and exports a function named **deletePost**.
* **id**: A parameter representing the ID of the post to be deleted.
* **axios.delete(${API\_URL}/${id})**: Uses Axios to send a DELETE request to the **API\_URL** with the specific **id**. This deletes the existing post.
* **return axios.delete(${API\_URL}/${id});**: Returns the Axios promise. This allows the caller to handle the asynchronous response (success or failure) using **.then()** and **.catch()** methods.

**2. Create the PostsAxios.js file**: This file will be the main component for managing posts.

// src/PostsAxios.js

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

import {createPost,  fetchPosts, updatePost, deletePost } from './PostServiceAxios';

const PostsAxios = () => {

  const [posts, setPosts] = useState([]);

  const [formData, setFormData] = useState({ title: '', userId: 1 });

  const [editMode, setEditMode] = useState(false);

  const [currentPostId, setCurrentPostId] = useState(null);

  useEffect(() => {

    fetchPosts()

      .then(response => {

        setPosts(response.data);

      })

      .catch(error => {

        console.error('Error fetching posts:', error);

      });

  }, []);

  const handleChange = (e) => {

    const { name, value } = e.target;

    setFormData({ ...formData, [name]: value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    if (editMode) {

      updatePost(currentPostId, formData)

        .then(response => {

          setPosts(posts.map(post => (post.id === currentPostId ? response.data : post)));

          setEditMode(false);

          setFormData({ title: '', userId: 1 });

        })

        .catch(error => {

          console.error('Error updating post:', error);

        });

    } else {

      createPost(formData)

        .then(response => {

          setPosts([...posts, response.data]);

          setFormData({ title: '', userId: 1 });

        })

        .catch(error => {

          console.error('Error creating post:', error);

        });

    }

  };

  const handleEdit = (post) => {

    setFormData({ title: post.title, userId: post.userId });

    setCurrentPostId(post.id);

    setEditMode(true);

  };

  const handleDelete = (id) => {

    deletePost(id)

      .then(() => {

        setPosts(posts.filter(post => post.id !== id));

      })

      .catch(error => {

        console.error('Error deleting post:', error);

      });

  };

  return (

    <div>

      <h2>{editMode ? 'Edit Post' : 'Create Post'}</h2>

      <form onSubmit={handleSubmit}>

        <div>

          <label>Title:</label>

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

        </div>

        <button type="submit">{editMode ? 'Update' : 'Create'}</button>

      </form>

      <h2>Posts</h2>

      <ul>

        {posts.map(post => (

          <li key={post.id}>

            <table><tr>

            <td>{post.title}</td>

            <td><button onClick={() => handleEdit(post)}>Edit</button></td>

            <td><button onClick={() => handleDelete(post.id)}>Delete</button></td>

            </tr></table>

          </li>

        ))}

      </ul>

    </div>

  );

};

export default PostsAxios;

Your **PostsAxios** component provides a functional React component that interacts with a REST API using Axios to perform CRUD operations.

Below is a detailed breakdown of each part of the code:

**1. Imports**

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

**import { createPost, fetchPosts, updatePost, deletePost } from './PostServiceAxios';**

* **React, useState, useEffect**: React and its hooks for managing state and side effects.
* **createPost, fetchPosts, updatePost, deletePost**: Functions imported from **PostServiceAxios** for interacting with the API.

**2. Component Definition**

**const PostsAxios = () => {**

* **PostsAxios**: Functional component definition.

**3. State Management**

**const [posts, setPosts] = useState([]);**

**const [formData, setFormData] = useState({ title: '', userId: 1 });**

**const [editMode, setEditMode] = useState(false);**

**const [currentPostId, setCurrentPostId] = useState(null);**

* **posts**: State to store the list of posts.
* **formData**: State to manage form input values.
* **editMode**: State to determine whether the form is in edit mode or create mode.
* **currentPostId**: State to store the ID of the post being edited.

**4. Fetch Posts on Mount**

**useEffect(() => {**

**fetchPosts()**

**.then(response => {**

**setPosts(response.data);**

**})**

**.catch(error => {**

**console.error('Error fetching posts:', error);**

**});**

**}, []);**

* **useEffect**: Hook to perform side effects in functional components. It runs once on component mount to fetch posts.
* **fetchPosts**: Fetches all posts from the API and updates the **posts** state with the response data.

**5. Handle Form Input Changes**

**const handleChange = (e) => {**

**const { name, value } = e.target;**

**setFormData({ ...formData, [name]: value });**

**};**

* **handleChange**: Updates **formData** state when form inputs change.

**6. Handle Form Submission**

**const handleSubmit = (e) => {**

**e.preventDefault();**

**if (editMode) {**

**updatePost(currentPostId, formData)**

**.then(response => {**

**setPosts(posts.map(post => (post.id === currentPostId ? response.data : post)));**

**setEditMode(false);**

**setFormData({ title: '', userId: 1 });**

**})**

**.catch(error => {**

**console.error('Error updating post:', error);**

**});**

**} else {**

**createPost(formData)**

**.then(response => {**

**setPosts([...posts, response.data]);**

**setFormData({ title: '', userId: 1 });**

**})**

**.catch(error => {**

**console.error('Error creating post:', error);**

**});**

**}**

**};**

* **handleSubmit**: Handles form submission. If in edit mode, it updates the post. Otherwise, it creates a new post.
* **e.preventDefault()**: Prevents the default form submission behavior.
* **updatePost**: Updates an existing post and modifies the **posts** state with the updated post.
* **createPost**: Creates a new post and appends it to the **posts** state.

**7. Handle Edit Action**

**const handleEdit = (post) => {**

**setFormData({ title: post.title, userId: post.userId });**

**setCurrentPostId(post.id);**

**setEditMode(true);**

**};**

* **handleEdit**: Populates the form with the post data to be edited and switches to edit mode.

**8. Handle Delete Action**

**const handleDelete = (id) => {**

**deletePost(id)**

**.then(() => {**

**setPosts(posts.filter(post => post.id !== id));**

**})**

**.catch(error => {**

**console.error('Error deleting post:', error);**

**});**

**};**

* **handleDelete**: Deletes a post and updates the **posts** state to remove the deleted post.

**9. JSX Return**

**return (**

**<div>**

**<h2>{editMode ? 'Edit Post' : 'Create Post'}</h2>**

**<form onSubmit={handleSubmit}>**

**<div>**

**<label>Title:</label>**

**<input type="text" name="title" value={formData.title} onChange={handleChange} />**

**</div>**

**<button type="submit">{editMode ? 'Update' : 'Create'}</button>**

**</form>**

**<h2>Posts</h2>**

**<ul>**

**{posts.map(post => (**

**<li key={post.id}>**

**<table>**

**<tr>**

**<td>{post.title}</td>**

**<td><button onClick={() => handleEdit(post)}>Edit</button></td>**

**<td><button onClick={() => handleDelete(post.id)}>Delete</button></td>**

**</tr>**

**</table>**

**</li>**

**))}**

**</ul>**

**</div>**

**);**

* **<h2>{editMode ? 'Edit Post' : 'Create Post'}</h2>**: Dynamic heading based on **editMode**.
* **<form onSubmit={handleSubmit}>**: Form for creating or editing a post.
* **<input type="text" name="title" value={formData.title} onChange={handleChange} />**: Controlled input for the post title.
* **<button type="submit">{editMode ? 'Update' : 'Create'}</button>**: Dynamic button text based on **editMode**.
* **<ul>**: List of posts.
* **{posts.map(post => ( ... ))}**: Maps over the **posts** array to render each post.
* **<button onClick={() => handleEdit(post)}>Edit</button>**: Button to edit a post.
* **<button onClick={() => handleDelete(post.id)}>Delete</button>**: Button to delete a post.

This React component provides a simple interface to manage posts using CRUD operations via the Axios library.

It handles fetching, creating, updating, and deleting posts while maintaining a responsive UI with form inputs and buttons.

**3. Create or update the App.js file**: This file will import and use the **Posts** component.

// src/App.js

// src/App.js

import React from 'react';

import './App.css'; // Import the CSS file

import PostsAxios from './PostsAxios';

const App = () => {

  return (

    <div>

      <h1>CRUD Application</h1>

      <PostsAxios />

    </div>

  );

};

export default App;

**Install Axios**: Make sure Axios is installed in your project. If it's not already installed, you can install it using npm.

**npm install axios@latest**

Here is a CSS file to style the components for the CRUD application. Create a file named **App.css** in your **src** directory and add the following styles:

/\* App.css \*/

/\* General Styles \*/

body {

  font-family: Arial, sans-serif;

  background-color: #f8f9fa;

  margin: 0;

  padding: 0;

}

h1 {

  text-align: center;

  color: #333;

  margin-top: 20px;

}

h2 {

  color: #333;

}

.container {

  max-width: 800px;

  margin: 0 auto;

  padding: 20px;

  background-color: #fff;

  border-radius: 5px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

  margin-top: 20px;

}

/\* Posts List \*/

ul {

  list-style: none;

  padding: 0;

}

li {

  display: flex;

  justify-content: space-between;

  align-items: center;

  padding: 10px;

  border-bottom: 1px solid #ddd;

}

li:last-child {

  border-bottom: none;

}

button {

  background-color: #007bff;

  color: white;

  border: none;

  padding: 5px 10px;

  border-radius: 3px;

  cursor: pointer;

  transition: background-color 0.3s ease;

}

button:hover {

  background-color: #0056b3;

}

button + button {

  margin-left: 5px;

}

/\* Form Styles \*/

form {

  margin-top: 20px;

}

form > div {

  margin-bottom: 10px;

}

label {

  display: block;

  margin-bottom: 5px;

}

input[type="text"] {

  width: 100%;

  padding: 8px;

  border: 1px solid #ccc;

  border-radius: 3px;

}

button[type="submit"] {

  background-color: #28a745;

}

button[type="submit"]:hover {

  background-color: #218838;

}

**Example 03**

**Let's rewrite the Example 02 program using the Fetch API instead of Axios.**

**1. Create the PostServiceFetch.js file**

This file will contain all the HTTP request functions using the Fetch API.

// src/PostServiceFetch.js

const API\_URL = 'https://jsonplaceholder.typicode.com/posts';

export const fetchPosts = async () => {

const response = await fetch(API\_URL);

if (!response.ok) {

throw new Error('Error fetching posts');

}

return response.json();

};

export const createPost = async (postData) => {

const response = await fetch(API\_URL, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(postData),

});

if (!response.ok) {

throw new Error('Error creating post');

}

return response.json();

};

export const updatePost = async (id, postData) => {

const response = await fetch(`${API\_URL}/${id}`, {

method: 'PUT',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify(postData),

});

if (!response.ok) {

throw new Error('Error updating post');

}

return response.json();

};

export const deletePost = async (id) => {

const response = await fetch(`${API\_URL}/${id}`, {

method: 'DELETE',

});

if (!response.ok) {

throw new Error('Error deleting post');

}

return response.json();

};

**2. Create the PostsFetch.css file**

/\* src/PostsFetch.css \*/

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

font-family: Arial, sans-serif;

background-color: #f9f9f9;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

border-radius: 8px;

}

h1, h2 {

text-align: center;

color: #333;

}

form {

display: flex;

flex-direction: column;

margin-bottom: 20px;

}

label {

margin-bottom: 5px;

font-weight: bold;

}

input[type="text"] {

padding: 10px;

margin-bottom: 10px;

border: 1px solid #ccc;

border-radius: 4px;

font-size: 16px;

}

button {

padding: 10px;

background-color: #007bff;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s;

}

button:hover {

background-color: #0056b3;

}

ul {

list-style: none;

padding: 0;

}

li {

display: flex;

justify-content: space-between;

align-items: center;

padding: 10px;

margin-bottom: 10px;

background-color: #fff;

border: 1px solid #ccc;

border-radius: 4px;

}

table {

width: 100%;

}

td {

padding: 5px;

}

td:first-child {

flex-grow: 1;

}

button.edit-btn {

background-color: #28a745;

margin-right: 5px;

}

button.edit-btn:hover {

background-color: #218838;

}

button.delete-btn {

background-color: #dc3545;

}

button.delete-btn:hover {

background-color: #c82333;

}

**3. Write PostsFetch.js**

Apply the CSS styles to the component by importing the CSS file.

// src/PostsFetch.js

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

import { createPost, fetchPosts, updatePost, deletePost } from './PostServiceFetch';

import './PostsFetch.css';

const PostsFetch = () => {

const [posts, setPosts] = useState([]);

const [formData, setFormData] = useState({ title: '', userId: 1 });

const [editMode, setEditMode] = useState(false);

const [currentPostId, setCurrentPostId] = useState(null);

useEffect(() => {

const getPosts = async () => {

try {

const data = await fetchPosts();

setPosts(data);

} catch (error) {

console.error('Error fetching posts:', error);

}

};

getPosts();

}, []);

const handleChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

};

const handleSubmit = async (e) => {

e.preventDefault();

try {

if (editMode) {

const updatedPost = await updatePost(currentPostId, formData);

setPosts(posts.map(post => (post.id === currentPostId ? updatedPost : post)));

setEditMode(false);

} else {

const newPost = await createPost(formData);

setPosts([...posts, newPost]);

}

setFormData({ title: '', userId: 1 });

} catch (error) {

console.error(`Error ${editMode ? 'updating' : 'creating'} post:`, error);

}

};

const handleEdit = (post) => {

setFormData({ title: post.title, userId: post.userId });

setCurrentPostId(post.id);

setEditMode(true);

};

const handleDelete = async (id) => {

try {

await deletePost(id);

setPosts(posts.filter(post => post.id !== id));

} catch (error) {

console.error('Error deleting post:', error);

}

};

return (

<div className="container">

<h1>CRUD Application</h1>

<h2>{editMode ? 'Edit Post' : 'Create Post'}</h2>

<form onSubmit={handleSubmit}>

<div>

<label htmlFor="title">Title:</label>

<input

type="text"

id="title"

name="title"

value={formData.title}

onChange={handleChange}

/>

</div>

<button type="submit">{editMode ? 'Update' : 'Create'}</button>

</form>

<h2>Posts</h2>

<ul>

{posts.map(post => (

<li key={post.id}>

<table>

<tr>

<td>{post.title}</td>

<td>

<button className="edit-btn" onClick={() => handleEdit(post)}>Edit</button>

</td>

<td>

<button className="delete-btn" onClick={() => handleDelete(post.id)}>Delete</button>

</td>

</tr>

</table>

</li>

))}

</ul>

</div>

);

};

export default PostsFetch;

**4. Create or update the App.js file**

This file will import and use the **PostsFetch** component.

// src/App.js

import React from 'react';

import './App.css'; // Import the CSS file

import PostsFetch from './PostsFetch';

const App = () => {

return (

<div>

<h1>CRUD Application</h1>

<PostsFetch />

</div>

);

};

export default App;

**Summary**

* **PostServiceFetch.js**: Contains functions for making HTTP requests using the Fetch API.
* **PostsFetch.js**: Main component for managing posts. It fetches posts on mount, handles form submissions for creating/updating posts, and handles edit/delete actions.
* **App.js**: Root component that imports and uses the **PostsFetch** component.

This setup provides a complete CRUD application using the Fetch API to interact with the JSONPlaceholder API.

**Example 04**

Below is a more comprehensive example of a React application that includes routing, CRUD operations, and a structured layout. The example uses **react-router-dom** for routing and **axios** for HTTP requests.

**Project Structure**

/react-app-012

/public

/src

/components

Navbar.js

PostList.js

PostForm.js

PostDetail.js

/services

PostService.js

App.js

index.js

App.css

**Dependencies**

First, make sure to install the necessary dependencies:

npm install react-router-dom@latest axios@latest

**PostService.js**

Create the **PostService.js** file for handling API requests.

// src/services/PostService.js

import axios from 'axios';

const API\_URL = 'https://jsonplaceholder.typicode.com/posts';

export const fetchPosts = () => {

return axios.get(API\_URL);

};

export const fetchPostById = (id) => {

return axios.get(`${API\_URL}/${id}`);

};

export const createPost = (postData) => {

return axios.post(API\_URL, postData);

};

export const updatePost = (postId, postData) => {

return axios.put(`${API\_URL}/${postId}`, postData);

};

export const deletePost = (postId) => {

return axios.delete(`${API\_URL}/${postId}`);

};

**Navbar.js**

Create a Navbar component for navigation.

// src/components/Navbar.js

import React from 'react';

import { Link } from 'react-router-dom';

import './Navbar.css';

const Navbar = () => {

return (

<nav className="navbar">

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/create">Create Post</Link></li>

</ul>

</nav>

);

};

export default Navbar;

**PostList.js**

Create a component to list all posts.

// src/components/PostList.js

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

import { Link } from 'react-router-dom';

import { fetchPosts, deletePost } from '../services/PostService';

import './PostList.css';

const PostList = () => {

const [posts, setPosts] = useState([]);

useEffect(() => {

fetchPosts()

.then(response => setPosts(response.data))

.catch(error => console.error('Error fetching posts:', error));

}, []);

const handleDelete = (id) => {

deletePost(id)

.then(() => setPosts(posts.filter(post => post.id !== id)))

.catch(error => console.error('Error deleting post:', error));

};

return (

<div className="post-list">

<h2>Posts</h2>

<ul>

{posts.map(post => (

<li key={post.id}>

<Link to={`/posts/${post.id}`}>{post.title}</Link>

<button onClick={() => handleDelete(post.id)}>Delete</button>

</li>

))}

</ul>

</div>

);

};

export default PostList;

**PostForm.js**

Create a component for creating and editing posts.

// src/components/PostForm.js

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

import { useHistory, useParams } from 'react-router-dom';

import { createPost, fetchPostById, updatePost } from '../services/PostService';

import './PostForm.css';

const PostForm = () => {

const [formData, setFormData] = useState({ title: '', body: '', userId: 1 });

const [editMode, setEditMode] = useState(false);

const { id } = useParams();

const history = useHistory();

useEffect(() => {

if (id) {

setEditMode(true);

fetchPostById(id)

.then(response => setFormData(response.data))

.catch(error => console.error('Error fetching post:', error));

}

}, [id]);

const handleChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

};

const handleSubmit = (e) => {

e.preventDefault();

const submitAction = editMode ? updatePost : createPost;

submitAction(id, formData)

.then(() => history.push('/'))

.catch(error => console.error('Error submitting form:', error));

};

return (

<div className="post-form">

<h2>{editMode ? 'Edit Post' : 'Create Post'}</h2>

<form onSubmit={handleSubmit}>

<div>

<label>Title:</label>

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

</div>

<div>

<label>Body:</label>

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

</div>

<button type="submit">{editMode ? 'Update' : 'Create'}</button>

</form>

</div>

);

};

export default PostForm;

**PostDetail.js**

Create a component to display a single post's details.

// src/components/PostDetail.js

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

import { useParams } from 'react-router-dom';

import { fetchPostById } from '../services/PostService';

import './PostDetail.css';

const PostDetail = () => {

const [post, setPost] = useState({});

const { id } = useParams();

useEffect(() => {

fetchPostById(id)

.then(response => setPost(response.data))

.catch(error => console.error('Error fetching post:', error));

}, [id]);

return (

<div className="post-detail">

<h2>{post.title}</h2>

<p>{post.body}</p>

</div>

);

};

export default PostDetail;

**App.js**

Set up the main application component with routing.

// src/App.js

import React from 'react';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

import Navbar from './components/Navbar';

import PostList from './components/PostList';

import PostForm from './components/PostForm';

import PostDetail from './components/PostDetail';

import './App.css';

const App = () => {

return (

<Router>

<div className="app">

<Navbar />

<div className="container">

<Switch>

<Route exact path="/" component={PostList} />

<Route exact path="/create" component={PostForm} />

<Route exact path="/posts/:id" component={PostDetail} />

<Route exact path="/edit/:id" component={PostForm} />

</Switch>

</div>

</div>

</Router>

);

};

export default App;

**index.js**

Render the **App** component.

// src/index.js

import React from 'react';

import ReactDOM from 'react-dom';

import App from './App';

import './index.css';

ReactDOM.render(

<React.StrictMode>

<App />

</React.StrictMode>,

document.getElementById('root')

);

**App.css**

Add some basic styling for the application.

/\* src/App.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

margin: 0;

padding: 0;

}

.app {

text-align: center;

}

.container {

max-width: 800px;

margin: 0 auto;

padding: 20px;

background-color: white;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

.navbar {

background-color: #333;

padding: 10px;

}

.navbar ul {

list-style: none;

padding: 0;

display: flex;

justify-content: center;

}

.navbar li {

margin: 0 10px;

}

.navbar a {

color: white;

text-decoration: none;

}

button {

background-color: #007bff;

color: white;

border: none;

padding: 10px;

border-radius: 5px;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #0056b3;

}

form {

display: flex;

flex-direction: column;

align-items: center;

}

form div {

margin-bottom: 10px;

}

form input {

padding: 10px;

width: 100%;

box-sizing: border-box;

}

form label {

display: block;

margin-bottom: 5px;

}

ul {

list-style: none;

padding: 0;

}

li {

padding: 10px;

border-bottom: 1px solid #ddd;

}

li:last-child {

border-bottom: none;

}

.post-detail {

text-align: left;

}

This example sets up a basic CRUD application with routing using React Router and Axios for HTTP requests. The application has routes for listing posts, creating new posts, editing posts, and viewing post details. The styling is kept simple for demonstration purposes, but you can expand and customize it as needed.

**Example 05**

Below is a more comprehensive guide for building an e-commerce project in React with routing, services, HTTP operations using Axios, and styled components using CSS.

We'll use React Router v6, which introduces some changes from v5.

**Step 1: Setup React Project**

First, create a new React project using Create React App.

**npx create-react-app ecommerce-app**

**cd ecommerce-app**

**npm start**

**Step 2: Install Dependencies**

Install **react-router-dom** for routing and **axios** for making HTTP requests.

**npm install react-router-dom@6 axios@latest**

**Step 3: Setup Project Structure**

Create a folder structure as follows:

src/

components/

Cart.js

Checkout.js

Home.js

Navbar.js

ProductDetail.js

ProductList.js

Cart.css

Checkout.css

Home.css

Navbar.css

ProductDetail.css

ProductList.css

pages/

HomePage.js

ProductListPage.js

ProductDetailPage.js

CartPage.js

CheckoutPage.js

services/

productService.js

cartService.js

orderService.js

userService.js

paymentService.js

App.js

index.js

App.css

**Step 4: Create Services with HTTP Operations**

**4.1. Product Service**

**src/services/productService.js**:

import axios from 'axios';

const API\_URL = 'https://dummyjson.com/products';

export const getProducts = async () => {

const response = await axios.get(API\_URL);

return response.data;

};

export const getProductById = async (id) => {

const response = await axios.get(`${API\_URL}/${id}`);

return response.data;

};

export const addProduct = async (product) => {

const response = await axios.post(API\_URL, product);

return response.data;

};

export const updateProduct = async (id, product) => {

const response = await axios.put(`${API\_URL}/${id}`, product);

return response.data;

};

export const deleteProduct = async (id) => {

const response = await axios.delete(`${API\_URL}/${id}`);

return response.data;

};

**Explanation:**

This code is a JavaScript module that interacts with a remote API to perform CRUD (Create, Read, Update, Delete) operations on products. The code uses the axios library to send HTTP requests. Below is an explanation of the code line by line:

**import axios from 'axios';**

* This line imports the **axios** library, which is used to make HTTP requests. It allows you to send requests to an API and handle the responses easily.

**const API\_URL = 'https://dummyjson.com/products';**

* This line defines a constant variable API\_URL that stores the base URL of the API endpoint for the products. The API endpoint provides various operations related to products.

**export const getProducts = async () => {**

**const response = await axios.get(API\_URL);**

**return response.data;**

**};**

* **getProducts**: This is an asynchronous function that retrieves a list of all products from the API.
  + **axios.get(API\_URL):** Sends a GET request to the API\_URL to fetch the products.
  + **response.data:** The response from the API is stored in the response object, and response.data contains the actual data returned by the API.
  + The function returns the data containing the list of products.

**export const getProductById = async (id) => {**

**const response = await axios.get(`${API\_URL}/${id}`);**

**return response.data;**

**};**

* **getProductById**: This is an asynchronous function that retrieves a single product by its ID.
  + **axios.get(${API\_URL}/${id}):** Sends a GET request to the API URL, appending the id of the product to fetch a specific product.
  + The function returns the data containing the details of the requested product.

**export const addProduct = async (product) => {**

**const response = await axios.post(API\_URL, product);**

**return response.data;**

**};**

* **addProduct**: This is an asynchronous function that adds a new product to the API.
  + **axios.post(API\_URL, product):** Sends a POST request to the API\_URL with the product object as the payload. The product object contains the data of the new product to be added.
  + The function returns the data containing the newly created product.

**export const updateProduct = async (id, product) => {**

**const response = await axios.put(`${API\_URL}/${id}`, product);**

**return response.data;**

**};**

* **updateProduct:** This is an asynchronous function that updates an existing product in the API.
  + **axios.put(${API\_URL}/${id}, product):** Sends a PUT request to the API URL with the id of the product to be updated and the product object containing the updated data.
  + The function returns the data containing the updated product details.

**export const deleteProduct = async (id) => {**

**const response = await axios.delete(`${API\_URL}/${id}`);**

**return response.data;**

**};**

* **deleteProduct**: This is an asynchronous function that deletes a product by its ID from the API.
  + **axios.delete(${API\_URL}/${id}):** Sends a DELETE request to the API URL, appending the id of the product to be deleted.
  + The function returns the data containing the result of the deletion operation (often a confirmation message or the deleted product data).

Overall, this module provides functions to interact with a products API, allowing for fetching all products, fetching a product by ID, adding a new product, updating an existing product, and deleting a product.

**4.2. Cart Service**

**src/services/cartService.js**:

let cart = [];

export const addToCart = (product) => {

cart.push(product);

return cart;

};

export const removeFromCart = (productId) => {

cart = cart.filter(product => product.id !== productId);

return cart;

};

export const getCartItems = () => {

return cart;

};

**Explanation:**

This code defines a simple shopping cart module in JavaScript that allows you toadd products to the cart, remove products from the cart, and retrieve the current items in the cart. Here’s a line-by-line explanation:

**let cart = [];**

* This line declares a variable cart as an empty array. The cart array will hold all the products that are added to the cart. Since it is defined outside any function, it serves as the module's shared storage for cart items.

**export const addToCart = (product) => {**

**cart.push(product);**

**return cart;**

**};**

* **addToCart**: This is an exported function that adds a product to the cart array.
  + **(product):** The function takes a product parameter, which is expected to be an object representing the product details.
  + **cart.push(product):** Adds the product to the end of the cart array.
  + **return** **cart**: Returns the updated cart array, now including the newly added product.

**export const removeFromCart = (productId) => {**

**cart = cart.filter(product => product.id !== productId);**

**return cart;**

**};**

* **removeFromCart**: This is an exported function that removes a product from the cart array based on its ID.
  + **(productId):** The function takes a productId parameter, which is the ID of the product to be removed.
  + **cart.filter(product => product.id !== productId):** This filters the cart array, keeping only the products whose id does not match the given productId. The cart variable is then reassigned to this filtered array, effectively removing the product with the matching ID.
  + **return cart**: Returns the updated cart array, now excluding the removed product.

**export const getCartItems = () => {**

**return cart;**

**};**

* **getCartItems**: This is an exported function that retrieves the current items in the cart.
  + return cart: Returns the cart array, which contains all the products currently in the cart.

**Summary**

* **cart** is an array that stores the products added to the cart.
* **addToCart** adds a product to the cart and returns the updated list of cart items.
* **removeFromCart** removes a product from the cart by its ID and returns the updated cart.
* **getCartItems** returns the current list of items in the cart.

This module allows for basic shopping cart functionality, such as adding and removing products and viewing the cart contents.

**4.3. Order Service**

**src/services/orderService.js**:

let orders = [];

export const placeOrder = (order) => {

orders.push(order);

return orders;

};

export const getOrders = () => {

return orders;

};

This code defines a simple module for managing orders in memory using JavaScript. Below is an explanation of the code line by line:

let orders = [];

* This line declares a variable orders as an empty array. The orders array will store all the orders that are placed. Since it is declared outside of any function, it acts as a shared storage for the entire module.

**export const placeOrder = (order) => {**

**orders.push(order);**

**return orders;**

**};**

* **placeOrder:** This is an exported function that adds a new order to the orders array.
  + **(order):** The function takes a parameter order, which is expected to be an object or any data structure representing an order.
  + **orders.push(order):** Adds the order to the end of the orders array.
  + return orders: Returns the updated orders array, now including the newly added order.

**export const getOrders = () => {**

**return orders;**

**};**

* **getOrders**: This is an exported function that retrieves the current list of all orders.
  + return orders: Returns the orders array, which contains all the orders that have been placed.

**Summary**

* orders is an array that stores all the orders.
* placeOrder allows you to add a new order to the orders array and returns the updated list of orders.
* getOrders returns the current list of orders.

This module essentially manages a simple in-memory order list, allowing for the addition and retrieval of orders.

**4.4. User Service**

**src/services/userService.js**:

let user = null;

export const login = (userInfo) => {

user = userInfo;

return user;

};

export const logout = () => {

user = null;

return user;

};

export const getUser = () => {

return user;

};

**4.5. Payment Service**

**src/services/paymentService.js**:

export const processPayment = (paymentInfo) => {

return { status: 'success', transactionId: '12345' };

};

**Step 5: Create Components**

**5.1. Navbar Component**

**src/components/Navbar.js**:

import React from 'react';

import { Link } from 'react-router-dom';

import './Navbar.css';

const Navbar = () => {

return (

<nav className="navbar">

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/products">Products</Link></li>

<li><Link to="/cart">Cart</Link></li>

</ul>

</nav>

);

};

export default Navbar;

**src/components/Navbar.css**:

.navbar {

background-color: #333;

padding: 1rem;

}

.navbar ul {

display: flex;

list-style: none;

margin: 0;

padding: 0;

}

.navbar ul li {

margin-right: 20px;

}

.navbar ul li a {

color: white;

text-decoration: none;

}

.navbar ul li a:hover {

text-decoration: underline;

}

**5.2. Home Component**

**src/components/Home.js**:

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

import { getProducts } from '../services/productService';

import './Home.css';

const Home = () => {

const [products, setProducts] = useState([]);

useEffect(() => {

const fetchProducts = async () => {

const data = await getProducts();

setProducts(data.products || data);

};

fetchProducts();

}, []);

return (

<div className="home">

<h1>Featured Products</h1>

<div className="product-list">

{products.slice(0, 4).map(product => (

<div key={product.id} className="product-item">

<h2>{product.title}</h2>

<p>${product.price}</p>

</div>

))}

</div>

</div>

);

};

export default Home;

**src/components/Home.css**:

.home {

text-align: center;

}

.product-list {

display: flex;

justify-content: center;

flex-wrap: wrap;

}

.product-item {

border: 1px solid #ccc;

margin: 10px;

padding: 20px;

width: 200px;

}

.product-item h2 {

font-size: 18px;

}

.product-item p {

font-size: 16px;

color: #333;

}

**5.3. ProductList Component**

**src/components/ProductList.js**:

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

import { getProducts } from '../services/productService';

import { Link } from 'react-router-dom';

import './ProductList.css';

const ProductList = () => {

const [products, setProducts] = useState([]);

useEffect(() => {

const fetchProducts = async () => {

const data = await getProducts();

setProducts(data.products || data);

};

fetchProducts();

}, []);

return (

<div className="product-list-page">

<h1>Products</h1>

<div className="product-list">

{products.map(product => (

<div key={product.id} className="product-item">

<h2>{product.title}</h2>

<p>${product.price}</p>

<Link to={`/products/${product.id}`} className="details-link">View Details</Link>

</div>

))}

</div>

</div>

);

};

export default ProductList;

**src/components/ProductList.css**:

.product-list-page {

text-align: center;

}

.product-list {

display: flex;

justify-content: center;

flex-wrap: wrap;

}

.product-item {

border: 1px solid #ccc;

margin: 10px;

padding: 20px;

width: 200px;

}

.product-item h2 {

font-size: 18px;

}

.product-item p {

font-size: 16px;

color: #333;

}

.details-link {

text-decoration: none;

color: #007bff;

}

.details-link:hover {

text-decoration: underline;

}

**5.4. ProductDetail Component**

**src/components/ProductDetail.js**:

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

import { useParams } from 'react-router-dom';

import { getProductById, deleteProduct, updateProduct } from '../services/productService';

import { addToCart } from '../services/cartService';

import './ProductDetail.css';

const ProductDetail = () => {

const { id } = useParams();

const [product, setProduct] = useState(null);

const [editMode, setEditMode] = useState(false);

const [productData, setProductData] = useState({ title: '', price: '' });

useEffect(() => {

const fetchProduct = async () => {

const data = await getProductById(id);

setProduct(data);

setProductData({ title: data.title, price: data.price });

};

fetchProduct();

}, [id]);

const handleDelete = async () => {

await deleteProduct(id);

alert('Product deleted successfully!');

};

const handleUpdate = async () => {

const updatedProduct = await updateProduct(id, productData);

setProduct(updatedProduct);

setEditMode(false);

alert('Product updated successfully!');

};

if (!product) return <div>Loading...</div>;

return (

<div className="product-detail">

{editMode ? (

<div className="edit-form">

<input

type="text"

value={productData.title}

onChange={(e) => setProductData({ ...productData, title: e.target.value })}

/>

<input

type="text"

value={productData.price}

onChange={(e) => setProductData({ ...productData, price: e.target.value })}

/>

<button onClick={handleUpdate}>Update</button>

</div>

) : (

<div className="product-info">

<h1>{product.title}</h1>

<p>${product.price}</p>

<button onClick={() => addToCart(product)}>Add to Cart</button>

</div>

)}

<button onClick={() => setEditMode(!editMode)}>

{editMode ? 'Cancel' : 'Edit'}

</button>

<button onClick={handleDelete}>Delete</button>

</div>

);

};

export default ProductDetail;

**src/components/ProductDetail.css**:

.product-detail {

text-align: center;

margin: 20px;

}

.product-info {

border: 1px solid #ccc;

padding: 20px;

}

.edit-form input {

margin: 5px;

padding: 5px;

}

button {

margin: 5px;

padding: 10px;

background-color: #007bff;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

**5.5. Cart Component**

**src/components/Cart.js**:

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

import { getCartItems, removeFromCart } from '../services/cartService';

import { Link } from 'react-router-dom';

import './Cart.css';

const Cart = () => {

const [cartItems, setCartItems] = useState([]);

useEffect(() => {

setCartItems(getCartItems());

}, []);

return (

<div className="cart">

<h1>Cart</h1>

<div className="cart-items">

{cartItems.map(item => (

<div key={item.id} className="cart-item">

<h2>{item.title}</h2>

<p>${item.price}</p>

<button onClick={() => setCartItems(removeFromCart(item.id))}>Remove</button>

</div>

))}

</div>

<Link to="/checkout" className="checkout-link">Proceed to Checkout</Link>

</div>

);

};

export default Cart;

**src/components/Cart.css**:

.cart {

text-align: center;

}

.cart-items {

display: flex;

flex-direction: column;

align-items: center;

}

.cart-item {

border: 1px solid #ccc;

margin: 10px;

padding: 20px;

width: 200px;

}

.cart-item h2 {

font-size: 18px;

}

.cart-item p {

font-size: 16px;

color: #333;

}

.checkout-link {

display: block;

margin-top: 20px;

text-decoration: none;

color: white;

background-color: #007bff;

padding: 10px;

}

.checkout-link:hover {

background-color: #0056b3;

}

**5.6. Checkout Component**

**src/components/Checkout.js**:

import React, { useState } from 'react';

import { getCartItems } from '../services/cartService';

import { placeOrder } from '../services/orderService';

import { processPayment } from '../services/paymentService';

import './Checkout.css';

const Checkout = () => {

const [cartItems] = useState(getCartItems());

const [orderStatus, setOrderStatus] = useState(null);

const handleCheckout = () => {

const order = { items: cartItems, total: cartItems.reduce((sum, item) => sum + item.price, 0) };

placeOrder(order);

const paymentInfo = { amount: order.total };

const paymentResult = processPayment(paymentInfo);

setOrderStatus(paymentResult.status);

};

return (

<div className="checkout">

<h1>Checkout</h1>

{cartItems.map(item => (

<div key={item.id} className="checkout-item">

<h2>{item.title}</h2>

<p>${item.price}</p>

</div>

))}

<button onClick={handleCheckout}>Place Order</button>

{orderStatus && <p>Order Status: {orderStatus}</p>}

</div>

);

};

export default Checkout;

**src/components/Checkout.css**:

.checkout {

text-align: center;

}

.checkout-item {

border: 1px solid #ccc;

margin: 10px;

padding: 20px;

width: 200px;

}

.checkout-item h2 {

font-size: 18px;

}

.checkout-item p {

font-size: 16px;

color: #333;

}

button {

margin: 5px;

padding: 10px;

background-color: #007bff;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

**Step 6: Setup Routing with React Router v6**

**6.1. HomePage Component**

**src/pages/HomePage.js**:

import React from 'react';

import Home from '../components/Home';

const HomePage = () => {

return (

<div>

<Home />

</div>

);

};

export default HomePage;

**6.2. ProductListPage Component**

**src/pages/ProductListPage.js**:

import React from 'react';

import ProductList from '../components/ProductList';

const ProductListPage = () => {

return (

<div>

<ProductList />

</div>

);

};

export default ProductListPage;

**6.3. ProductDetailPage Component**

**src/pages/ProductDetailPage.js**:

import React from 'react';

import ProductDetail from '../components/ProductDetail';

const ProductDetailPage = () => {

return (

<div>

<ProductDetail />

</div>

);

};

export default ProductDetailPage;

**6.4. CartPage Component**

**src/pages/CartPage.js**:

import React from 'react';

import Cart from '../components/Cart';

const CartPage = () => {

return (

<div>

<Cart />

</div>

);

};

export default CartPage;

**6.5. CheckoutPage Component**

**src/pages/CheckoutPage.js**:

import React from 'react';

import Checkout from '../components/Checkout';

const CheckoutPage = () => {

return (

<div>

<Checkout />

</div>

);

};

export default CheckoutPage;

**Step 7: Assemble Everything in App.js**

**src/App.js**:

import React from 'react';

import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';

import HomePage from './pages/HomePage';

import ProductListPage from './pages/ProductListPage';

import ProductDetailPage from './pages/ProductDetailPage';

import CartPage from './pages/CartPage';

import CheckoutPage from './pages/CheckoutPage';

import Navbar from './components/Navbar';

import './App.css';

function App() {

return (

<Router>

<div className="App">

<Navbar />

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="products" element={<ProductListPage />} />

<Route path="products/:id" element={<ProductDetailPage />} />

<Route path="cart" element={<CartPage />} />

<Route path="checkout" element={<CheckoutPage />} />

</Routes>

</div>

</Router>

);

}

export default App;

**Step 8: Add Global Styling**

For global styling, you can add styles in **src/App.css**.

**src/App.css**:

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f8f9fa;

}

.App {

text-align: center;

}

h1 {

color: #333;

}

h2 {

color: #555;

}

button {

cursor: pointer;

}

**Step 9: Run the Project**

Run the project to see your e-commerce application in action.

**npm start**

You should now have a comprehensive e-commerce application with routing, services, HTTP operations using Axios, and styled components.

This application allows users to view products, add them to the cart, and proceed to checkout.

The services handle product data fetching, cart operations, order processing, user management, and payment simulation.