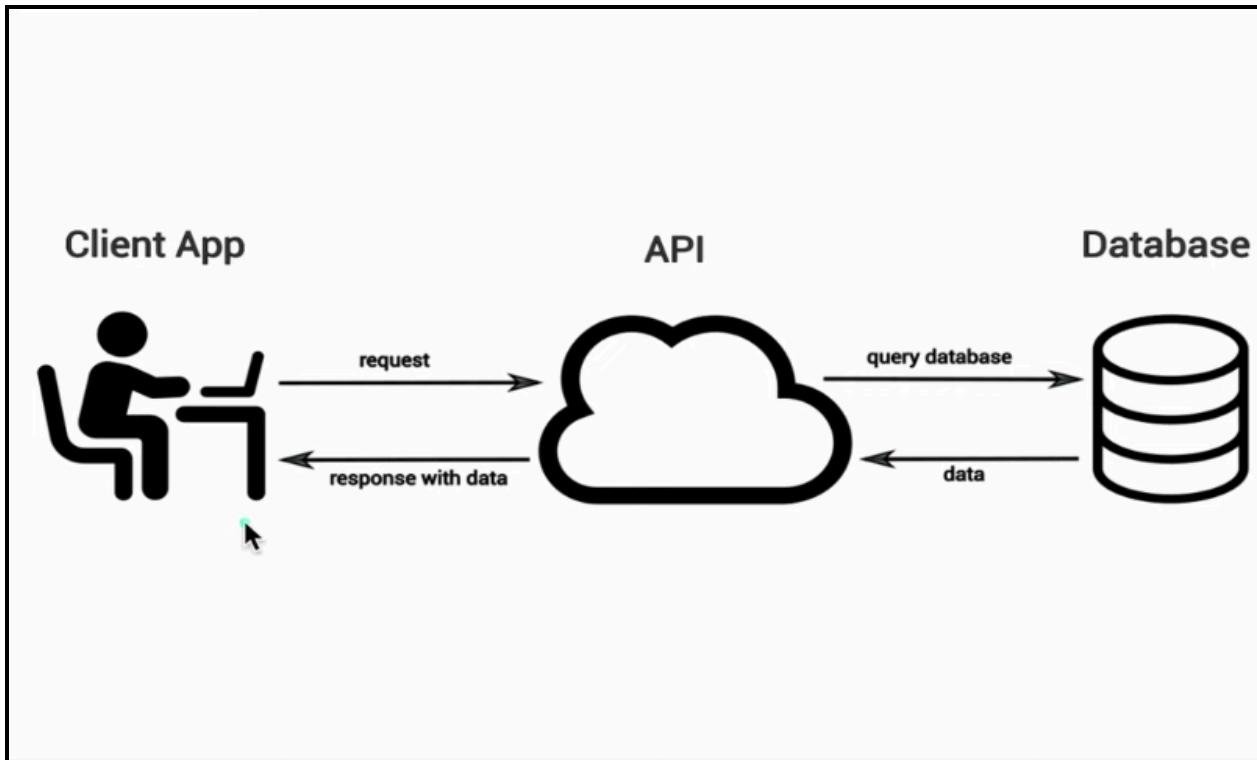


API CALLING IN REACT JS

1: What is API

API stands for Application Programming Interface

Fetching Data From Network



=> **{JSON} Placeholder:** Free fake API For testing and prototyping.

Resources

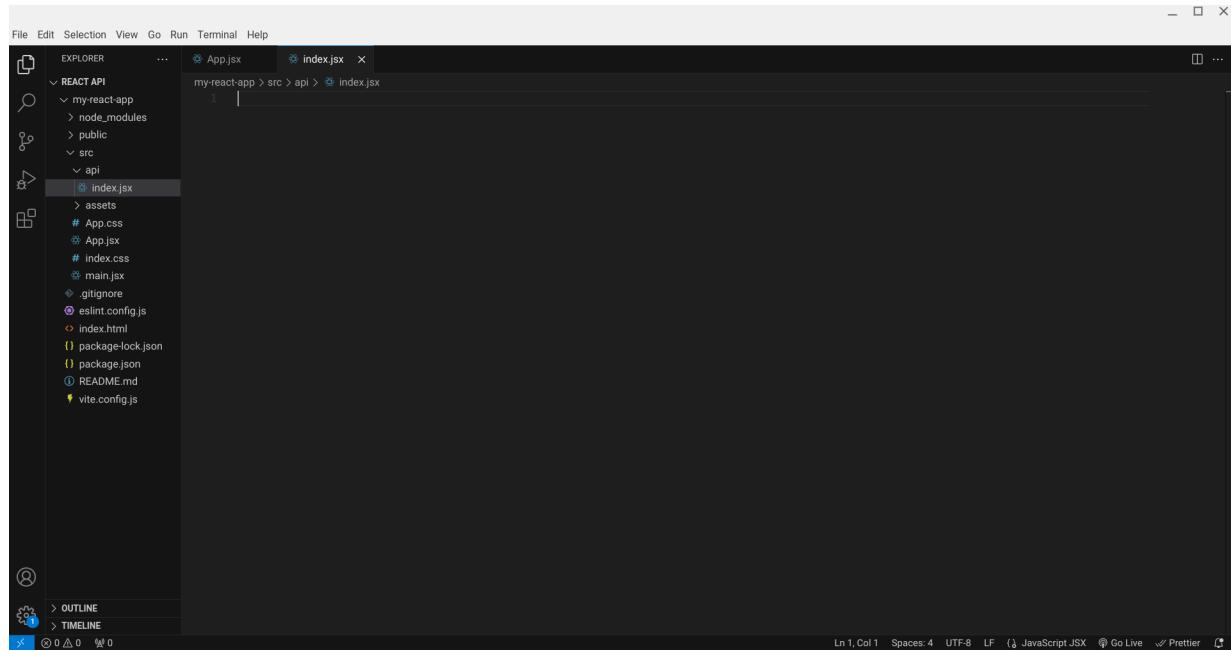
JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	1000 todos
/users	10 users

Note: resources have relations. For example: posts have many comments, albums have many photos, ... see [guide](#) for the full list.

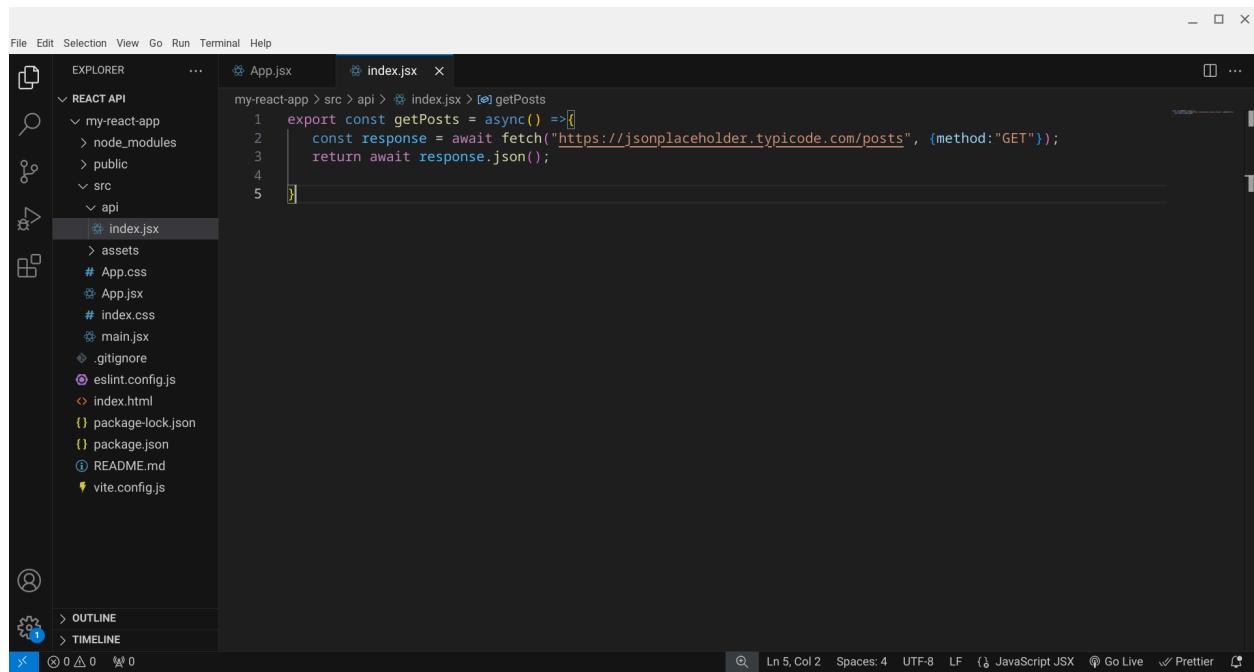
First Of all late create a api folder in src folder then create index.jsx file in api filder

Show in figure:



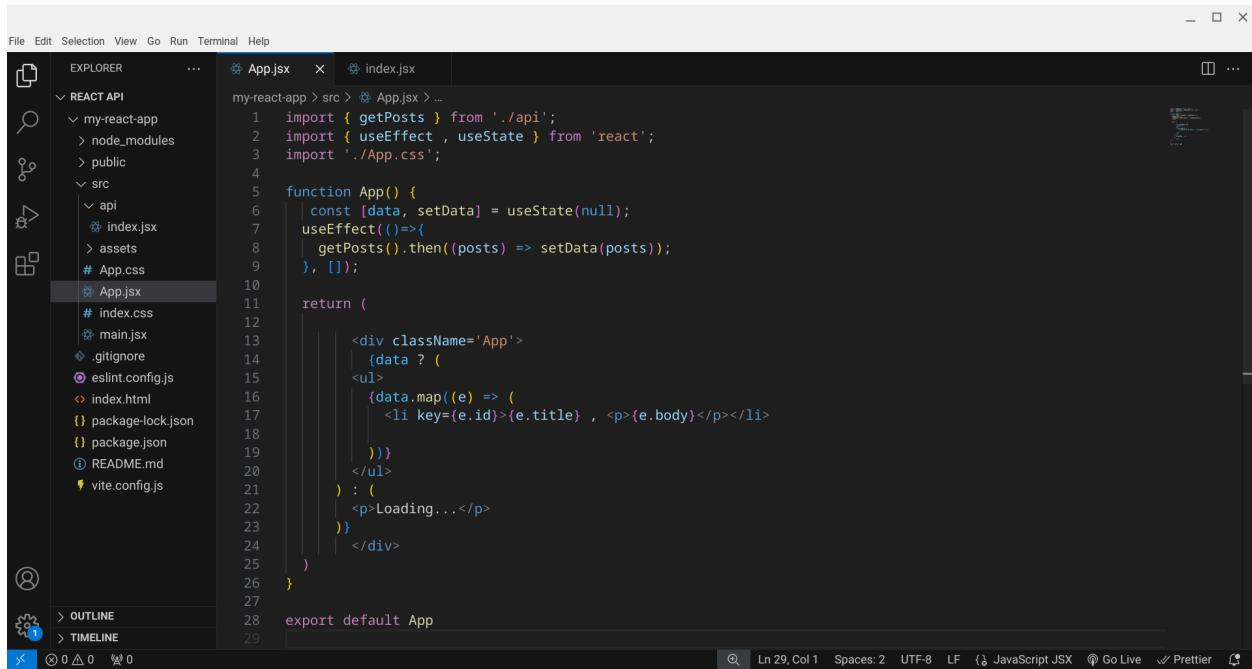
Now create fetch code in index.jsx file

Show in figure:



Now call In App.jsx

In React, useEffect is used to run side effects like API calls when a component mounts. useState stores the fetched data so the component can re-render automatically. We use fetch with async/await to get data from an API asynchronously. Data is displayed in JSX, often with conditional rendering to handle loading states. This flow ensures data is fetched once and shown dynamically in the UI.



The screenshot shows a code editor interface with the following details:

- File Explorer:** On the left, it shows a project structure for "my-react-app". The "src" folder contains "api", "index.jsx", "assets", "# App.css", and "# App.jsx".
- Code Editor:** The main area displays the content of "App.jsx".

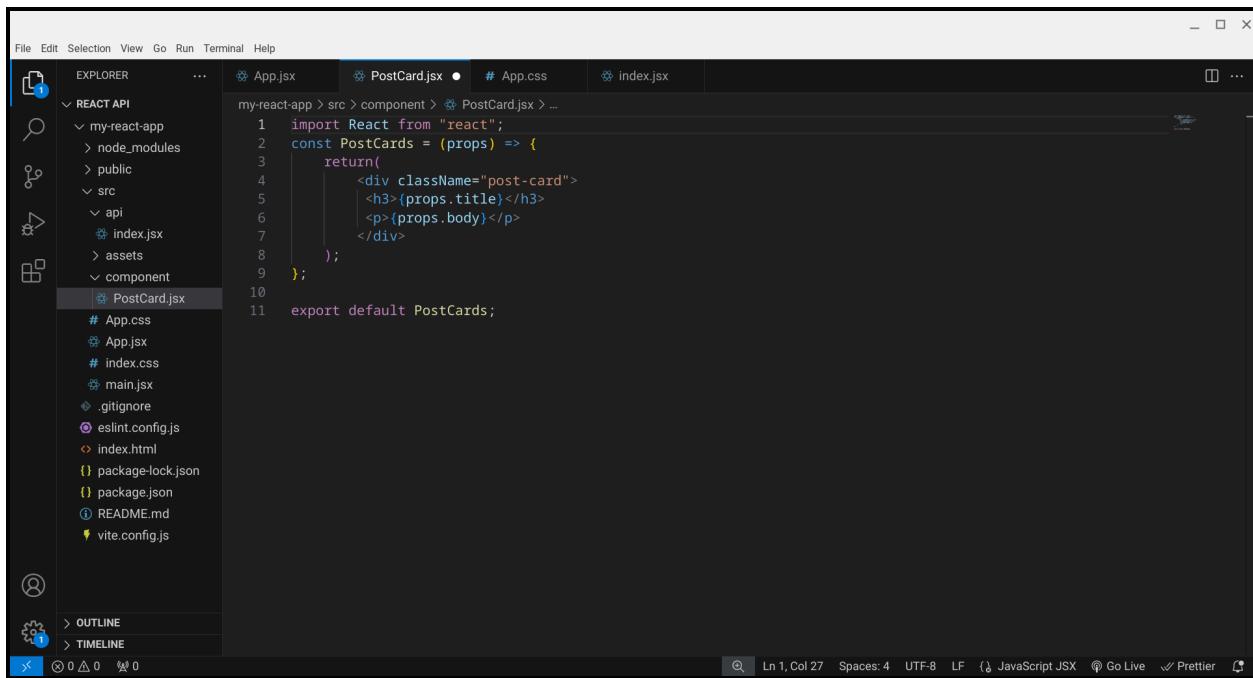
```
my-react-app > src > App.jsx > ...
1  import { getPosts } from './api';
2  import { useEffect, useState } from 'react';
3  import './App.css';
4
5  function App() {
6    const [data, setData] = useState(null);
7    useEffect(()=>{
8      getPosts().then((posts) => setData(posts));
9    }, []);
10
11   return (
12     <div className='App'>
13       {data ? (
14         <ul>
15           {data.map((e) => (
16             <li key={e.id}>{e.title} , <p>{e.body}</p></li>
17           )));
18         </ul>
19       ) : (
20         <p>Loading...</p>
21       )
22     </div>
23   )
24 }
25
26 }
27
28 export default App
```
- Status Bar:** At the bottom, it shows "Ln 29, Col 1" and other settings like "Spaces: 2", "UTF-8", "LF", "JavaScript JSX", "Go Live", "Prettier", and a refresh icon.

Now Create component Folder in src then create PostCard.jsx in component file

In React, we can pass data from a parent component to a child component using **props**.

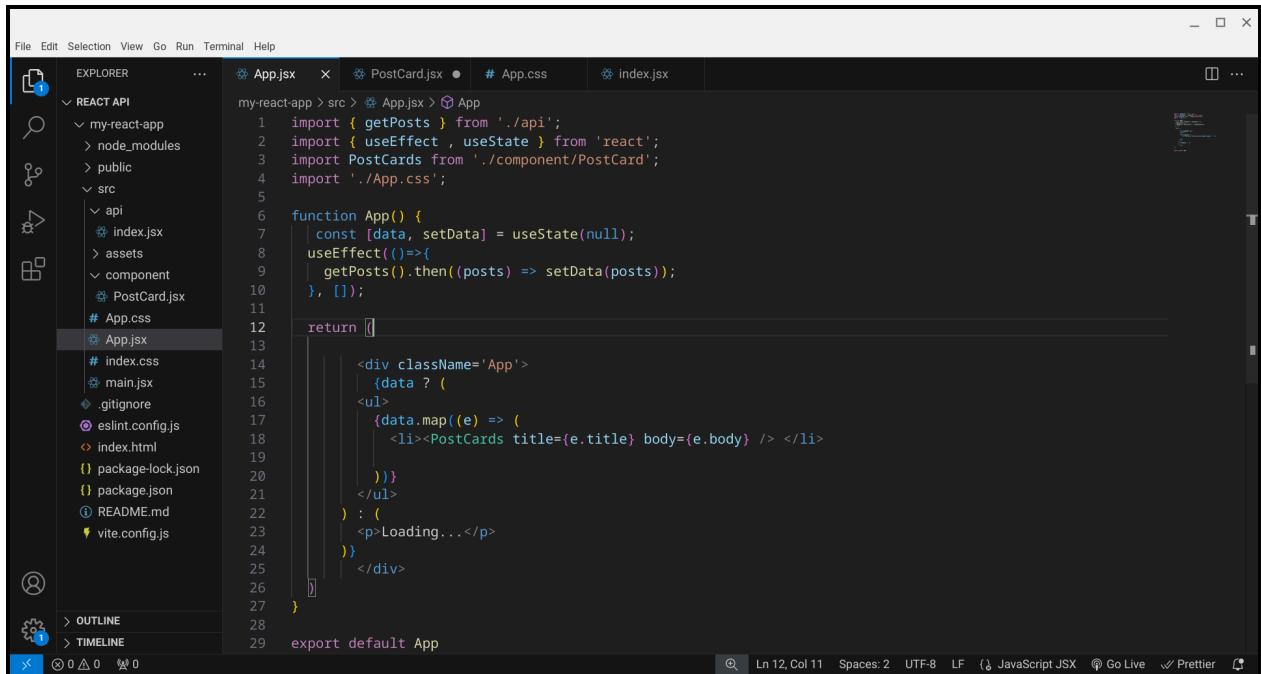
The parent fetches or stores the data, and then **sends it down** to the child as a prop.

The child component can access this data and display it, allowing us to **keep components reusable and organized**. This approach makes the UI cleaner and easier to style, as each component is responsible for **rendering its own piece of data**. By combining **props with conditional rendering**, we can design professional and dynamic layouts that handle loading or empty states gracefully.



```
my-react-app > src > component > PostCard.jsx > ...
1  import React from "react";
2  const PostCards = (props) => {
3      return(
4          <div className="post-card">
5              <h3>{props.title}</h3>
6              <p>{props.body}</p>
7          </div>
8      );
9  };
10 export default PostCards;
```

We can pass data from a child component up to the parent component in React using props callbacks. The child component receives a function prop from the parent and calls it, sending its data as an argument. This allows the parent component (`App.jsx`) to receive and store the data, enabling it to control or display it as needed. Using this pattern keeps the components decoupled, reusable, and easier to manage. It is especially useful for dynamic interactions, where the child generates or updates data that the parent needs to use display in the UI:



```

File Edit Selection View Go Run Terminal Help
EXPLORER ... App.jsx PostCard.jsx # App.css index.jsx
REACT API
my-react-app > src > App.jsx > App
1 import { getPosts } from './api';
2 import { useEffect, useState } from 'react';
3 import PostCards from './Component/PostCard';
4 import './App.css';
5
6 function App() {
7   const [data, setData] = useState(null);
8   useEffect(()=>{
9     getPosts().then((posts) => setData(posts));
10   }, []);
11
12   return [
13     <div className='App'>
14       <ul>
15         {data ? (
16           <ul>
17             {data.map((e) => (
18               <li><PostCards title={e.title} body={e.body}>/</li>
19             ))
20           </ul>
21         ) : (
22           <p>Loading...</p>
23         )}
24       </div>
25     ]
26   }
27 }
28
29 export default App
Ln 12, Col 11 Spaces: 2 UTF-8 LF { JavaScript JSX ⚡ Go Live ✨ Prettier

```

Now the same Api In axios Api In react js

First of All install: `npm install axios`

Create `api.jsx` (to organize API calls)

```

// src/api.jsx
import axios from "axios";

// Function to fetch posts
export const getPosts = async ()=> {
  try {
    const response = await axios.get("https://jsonplaceholder.typicode.com/posts");
    return response.data;
  } catch (error) {
    throw error;
  }
};

```

Then Create Child Component Posts.jsx

```
// src/Posts.jsx
import React from "react";

function Posts({ posts }) {
  return (
    <div>
      <h2>Posts List</h2>
      <ul>
        {posts.map((post) => (
          <li key={post.id}>
            <strong>{post.title}</strong>
            <p>{post.body}</p>
          </li>
        )))
      </ul>
    </div>
  );
}

export default Posts;
```

Parent Component App.jsx

```
// src/App.jsx
import React, { useEffect, useState } from "react";
import { getPosts } from "./api";
import Posts from "./Posts";
import "./App.css";

function App() {
  const [posts, setPosts] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchPosts = async () => {
      try {
        const data = await getPosts();
        setPosts(data);
      } catch (err) {
        setError("Failed to fetch posts");
      } finally {
        setLoading(false);
      }
    }
  }, []);
```

```

    };

    fetchPosts();
}, []);

return (
  <div className="App">
    <h1>React Axios API Example</h1>

    {loading && <p>Loading...</p>}
    {error && <p style={{ color: "red" }}>{error}</p>}
    {!loading && !error && <Posts posts={posts} />}
  </div>
);
}

export default App;

```

Key Points

1. `axios.get()` → fetch API data
2. `try/catch/finally` → handles errors and loading state
3. `useState` → stores posts, loading, error
4. `useEffect` → fetches data once on component mount
5. Data is passed to **child component via props** → keeps code reusable and organized