

Internship Report – Frontend Dev

Week 7: React.js Advanced

Name: Zainab

Father Name: Assad Qayyum

Date: 8th Aug, 2025

Internship Domain: Front-end Intern

Task: Mini Project - The Blog Viewer using API

Task Overview: (Day5)

The task for today was to build a Mini Project in React.js that applies all the concepts learned so far in Week 6 (React Basics) and Week 7 (React Advanced). I have selected the project **Blog Viewer** that fetches and displays real blog posts from an API.

Objective:

The objective was:

- To implement concepts of react.js such as routing, API calls etc.
- To understand and use real APIs for fetching live data.
- To practice component-based architecture and state management in React.
- To enhance the project's styling and UI for a better user experience.

Mini-project: Blog Viewer using API

The Blog Viewer is a simple React.js application that fetches real blog posts from the Sling Academy API and displays them in a user-friendly interface.

The home page lists all available blog posts, showing the title, an image (if available), and a short excerpt of the content. Each post has a “Read More” link that navigates to a detailed view page showing the full content of the blog. The project uses React Router for navigation, a custom useFetch hook for API calls, and CSS for styling the interface.

All the Concepts Used

1. React Router

Used for navigating between pages:

- / → Displays the list of blog posts.
- /post/:id → Displays details of a selected post.

2. useEffect for API Calls

Used to fetch real blog data from the Sling Academy API when the component mounts.

3. Custom Hook (useFetch)

Created a reusable hook to fetch data from any given API endpoint and handle loading/error states.

4. React Project Folder Structure

Organized into folders:

- components/ for UI components like PostList and PostDetail.
- hooks/ for reusable logic (useFetch.js).

5. Conditional Styling

Used CSS classes to make posts and details look clean and responsive.

Future Improvements:

While the current version of the Blog Viewer is functional and visually appealing, there are several enhancements that could improve user experience and functionality in future iterations:

- **Search Bar:** Allow users to search for blog posts by title or keyword to quickly find relevant articles.
- **Pagination or Infinite Scroll:** Instead of loading all posts at once, implement pagination or infinite scroll for better performance and smoother browsing.
- **Dark Mode Toggle:** Add a theme switcher to let users choose between light and dark mode for better accessibility.
- **Category Filters:** Enable filtering of posts based on categories or tags for more organized content navigation.

Practice Code:

This Blog Viewer is a simple React.js app that fetches blog posts from a public API and displays them in a clean, styled interface.

- Fetches and displays blog posts from a public API (JSONPlaceholder).
- Uses React Router for multi-page navigation.
- Implements Context API for light/dark theme toggle.
- Includes a custom hook (useFetch) for API calls.
- Applies conditional styling for different themes.

Index.js:

```
JS index.js X JS App.js JS ThemeContext.js JS useFetch.js JS Navbar.js JS PostList.js JS Home.js JS PostDe ▶ □ ...
src > JS index.js
1 import React from "react";
2 import { createRoot } from "react-dom/client";
3 import { BrowserRouter } from "react-router-dom";
4 import App from "../App";
5 import { ThemeProvider } from "../context/ThemeContext";
6 import "../index.css";
7
8 createRoot(document.getElementById("root")).render(
9   <React.StrictMode>
10     <BrowserRouter>
11       <ThemeProvider>
12         <App />
13       </ThemeProvider>
14     </BrowserRouter>
15   </React.StrictMode>
16 );
17
```

App.js:

```
JS index.js JS App.js X JS ThemeContext.js JS useFetch.js JS Navbar.js JS PostList.js JS Home.js JS PostDe > < ...
src > JS App.js > ...
1 import React from "react";
2 import { Routes, Route } from "react-router-dom";
3 import Navbar from "../components/Navbar";
4 import Home from "../pages/Home";
5 import PostDetail from "../pages/PostDetail";
6
7 function App() {
8   return (
9     <div>
10       <Navbar />
11       <Routes>
12         <Route path="/" element={ <Home /> } />
13         <Route path="/posts/:id" element={ <PostDetail /> } />
14       </Routes>
15     </div>
16   );
17 }
18
19 export default App;
20
```

ThemeContext.js:

```
JS index.js JS App.js JS ThemeContext.js X JS useFetch.js JS Navbar.js JS PostList.js JS Home.js JS PostDe > < ...
src > context > JS ThemeContext.js > ...
1 import React, { createContext, useState } from "react";
2
3 export const ThemeContext = createContext();
4
5 export function ThemeProvider({ children }) {
6   const [theme, setTheme] = useState("light");
7
8   const toggleTheme = () => {
9     setTheme(prev => prev === "light" ? "dark" : "light");
10   };
11
12   return (
13     <ThemeContext.Provider value={{ theme, toggleTheme }}>
14       <div className={theme === "light" ? "light-theme" : "dark-theme"}>
15         {children}
16       </div>
17     </ThemeContext.Provider>
18   );
19 }
20
```

useFetch.js:

```
JS index.js JS App.js JS ThemeContext.js JS useFetch.js X JS Navbar.js JS PostList.js JS Home.js JS PostDe > < ...
src > hooks > JS useFetch.js > ...
1 import { useState, useEffect } from "react";
2
3 export default function useFetch(url) {
4   const [data, setData] = useState(null);
5   const [loading, setLoading] = useState(true);
6
7   useEffect(() => {
8     fetch(url)
9       .then(res => res.json())
10      .then(data => {
11        setData(data);
12        setLoading(false);
13      })
14      .catch(() => setLoading(false));
15   }, [url]);
16
17   return { data, loading };
18 }
19
```

Navbar.js:

```
JS index.js JS App.js JS ThemeContext.js JS useFetch.js JS Navbar.js X JS PostList.js JS Home.js JS PostDe
src > components > JS Navbar.js > ...
1 import React, { useContext } from "react";
2 import { Link } from "react-router-dom";
3 import { ThemeContext } from "../context/ThemeContext";
4
5 function Navbar() {
6   const { theme, toggleTheme } = useContext(ThemeContext);
7
8   return (
9     <nav className="navbar">
10       <Link to="/" className="nav-title">Blog Viewer</Link>
11       <button onClick={toggleTheme}>
12         {theme === "light" ? "Dark Mode" : "Light Mode"}
13       </button>
14     </nav>
15   );
16 }
17
18 export default Navbar;
19
```

PostList.js:

```
JS index.js JS App.js JS ThemeContext.js JS useFetch.js JS Navbar.js JS PostList.js X JS Home.js JS PostDe
src > components > JS PostList.js > PostList
1 import React from "react";
2 import { Link } from "react-router-dom";
3 import useFetch from "../hooks/useFetch";
4
5 function PostList() {
6   const { data: posts, loading } = useFetch("https://jsonplaceholder.typicode.com/posts");
7
8   if (loading) return <p>Loading...</p>;
9
10   return (
11     <div className="post-list">
12       {posts.slice(0, 10).map(post => (
13         <div key={post.id} className="post-card">
14           <h3>{post.title}</h3>
15           <Link to={`/posts/${post.id}`}>Read More</Link>
16         </div>
17       ))}
18     </div>
19   );
20 }
21
22 export default PostList;
23
```

Home.js:

```
JS index.js JS App.js JS ThemeContext.js JS useFetch.js JS Navbar.js JS PostList.js JS Home.js X JS PostDe
src > pages > JS Home.js > ...
1 import React from "react";
2 import PostList from "../components/PostList";
3
4 function Home() {
5   return (
6     <div className="container">
7       <h1>Latest Posts</h1>
8       <PostList />
9     </div>
10   );
11 }
12
13 export default Home;
14
```

PostDetail.js:

```
JS App.js JS ThemeContext.js JS useFetch.js JS Navbar.js JS PostList.js JS Home.js JS PostDetail.js X # i > [] ...
src > pages > JS PostDetail.js > ...
1 import React from "react";
2 import { useParams, Link } from "react-router-dom";
3 import useFetch from "../hooks/useFetch";
4
5 function PostDetail() {
6   const { id } = useParams();
7   const { data: post, loading } = useFetch(`https://jsonplaceholder.typicode.com/posts/${id}`);
8
9   if (loading) return <p>Loading...</p>;
10
11   return (
12     <div className="container">
13       <Link to="/">← Back</Link>
14       <h2>{post.title}</h2>
15       <p>{post.body}</p>
16     </div>
17   );
18 }
19
20 export default PostDetail;
21
```

Index.css:

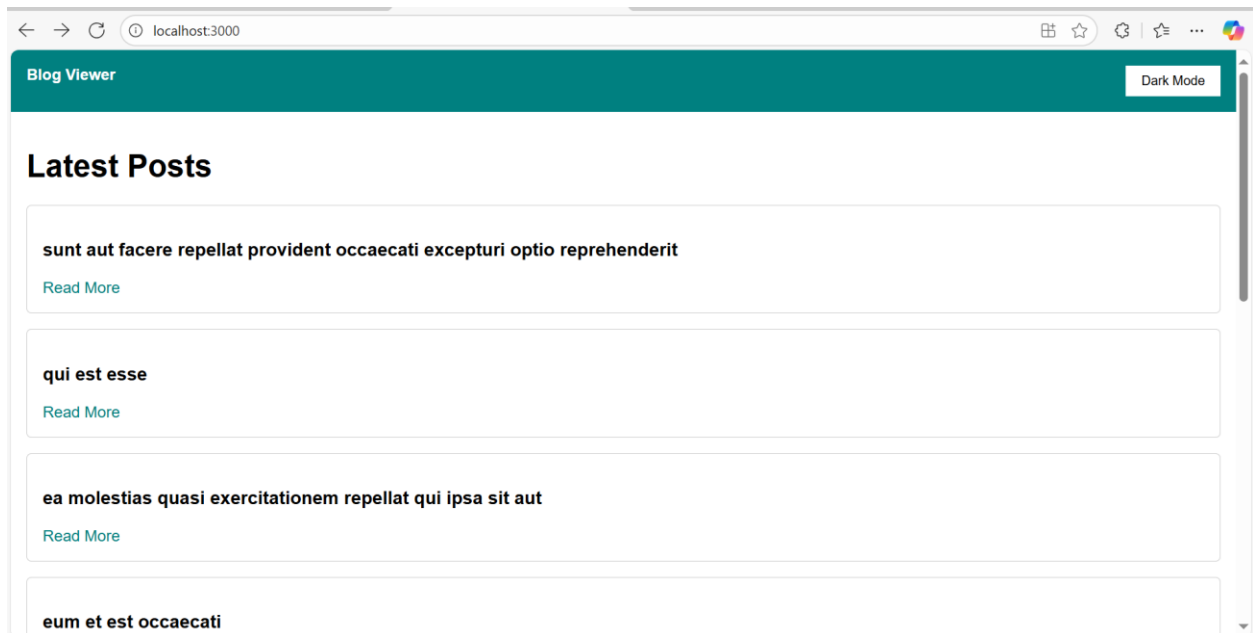
```
src > # index.css > .post-card
1 body {
2   margin: 0;
3   font-family: Arial, sans-serif;
4 }
5
6 .light-theme {
7   background-color: #fff;
8   color: #000;
9 }
10
11 .dark-theme {
12   background-color: #1e1e1e;
13   color: #fff;
14 }
15
16 .navbar {
17   display: flex;
18   justify-content: space-between;
19   padding: 1rem;
20   background-color: teal;
21   color: white;
22 }
23
24 .nav-title {
25   text-decoration: none;
26   color: white;
27   font-weight: bold;
28 }
29
30 button {
```

Ln 54, Col 2 Spaces: 2 UTF-8 LF () CSS Go Live

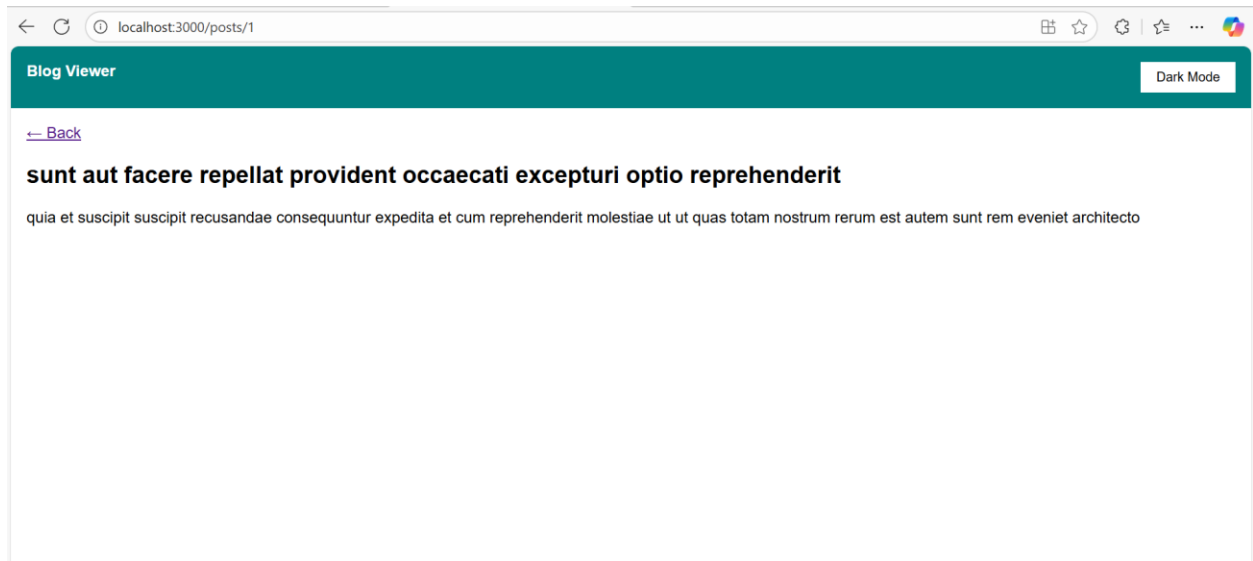
```
src > # index.css > .post-card

30 button {
31   background: white;
32   border: none;
33   padding: 0.5rem 1rem;
34   cursor: pointer;
35 }
36
37 button:hover {
38   opacity: 0.8;
39 }
40
41 .container {
42   padding: 1rem;
43 }
44
45 .post-list {
46   display: grid;
47   gap: 1rem;
48 }
49
50 .post-card {
51   padding: 1rem;
52   border: 1px solid #ccc;
53   border-radius: 5px;
54 }
55 .post-card a {
56   color: teal;
57   text-decoration: none;
58 }
59
```

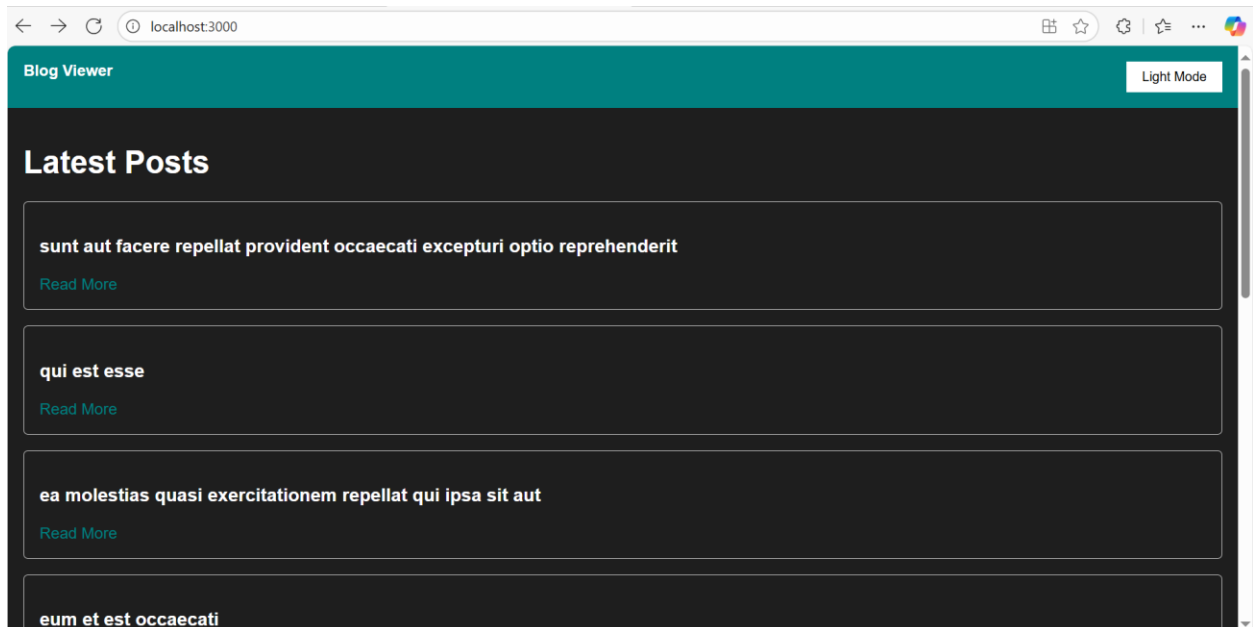
Ln 54, Col 2 Spaces: 2 UTF-8 LF {} CSS Go Live



Read more for post1:



Dark mode:



Conclusion:

Today, **The Blog Viewer using API** project successfully demonstrated the ability to work with React.js advanced concepts while integrating real-world data through an API. This mini-project enhanced understanding of routing, hooks, and API integration in React.