🧭Day 21: Mini Project - Simple Blog App with Routing + Context

🚢Goal

To combine everything learned so far — components, props, state management, context, routing, and local storage — into a small but functional **Blog App**.

Key Concepts Recap

1. Routing

Routing allows us to navigate between pages without refreshing the browser. We use `react-router-dom` to handle it.

Example:

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
import Home from './Components/Home';
import Blogs from './Components/Blogs';
import AddBlog from './Components/AddBlog';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/blogs" element={<Blogs />} />
        <Route path="/add-blog" element={<AddBlog />} />
      </Routes>
    </BrowserRouter>
  );
}
```

2. Context API

The Context API allows you to share state across multiple components without prop drilling.

Example:

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

export const BlogContext = createContext();

export const BlogProvider = ({ children }) => {
  const [posts, setPosts] = useState(() => {
    const localData = localStorage.getItem('posts');
    return localData ? JSON.parse(localData) : [];
```

```
  });

  useEffect(() => {
    localStorage.setItem('posts', JSON.stringify(posts));
  }, [posts]);

  return (
    <BlogContext.Provider value={{ posts, setPosts }}>
      {children}
    </BlogContext.Provider>
  );
};
```

3. Local Storage

We store the blog posts in the browser's local storage so that data remains even after page refresh.

```
useEffect(() => {
  localStorage.setItem('posts', JSON.stringify(posts));
}, [posts]);
```

4. Adding and Displaying Posts

We'll create an input form to add new posts and display them dynamically.

🏗️Project Structure

```
/src
│
├── /Components
│   ├── Home.jsx
│   ├── Blogs.jsx
│   ├── AddBlog.jsx
│   └── BlogDetails.jsx
│
├── /BlogContext
│   └── BlogPosts.jsx
│
└── App.jsx
```

Step-by-Step Project Flow

1. **Create Blog Context** — to store and manage posts.
2. **Wrap App in Blog Context Provider**.

3. **Create Routes:**
4. `/` → Home Page
5. `/blogs` → All Blogs
6. `/add-blog` → Add New Blog
7. `/blog_details/:index` → Blog Detail Page
8. **Use** `useContext` to access and modify post data.
9. **Store posts in Local Storage** for persistence.
10. **Display individual blog details using params.**

🏠Example Component: AddBlog.jsx

```
import React, { useState, useContext } from 'react';
import { BlogContext } from '../BlogContext/BlogPosts';
import { useNavigate } from 'react-router-dom';

export default function AddBlog() {
  const [title, setTitle] = useState('');
  const [content, setContent] = useState('');
  const { posts, setPosts } = useContext(BlogContext);
  const navigate = useNavigate();

  const handleSubmit = (e) => {
    e.preventDefault();
    setPosts([...posts, { title, content }]);
    navigate('/blogs');
  };

  return (
    <form onSubmit={handleSubmit}>
      <h3>Add a New Blog</h3>
      <input
        type="text"
        placeholder="Title"
        value={title}
        onChange={(e) => setTitle(e.target.value)}
        required
      />
      <textarea
        placeholder="Content"
        value={content}
        onChange={(e) => setContent(e.target.value)}
        required
      />
      <button type="submit">Add Blog</button>
    </form>
  );
}
```

Example Component: Blogs.jsx

```
import React, { useContext } from 'react';
import { BlogContext } from '../BlogContext/BlogPosts';
import { Link } from 'react-router-dom';

export default function Blogs() {
  const { posts } = useContext(BlogContext);

  return (
    <div>
      <h2>All Blogs</h2>
      {posts.length === 0 ? (
        <p>No blogs yet! Add one.</p>
      ) : (
        posts.map((post, index) => (
          <div key={index}>
            <h3>{post.title}</h3>
            <Link to={`/blog_details/${index}`}>Read More</Link>
          </div>
        ))
      )}
    </div>
  );
}
```

🧭BlogDetails.jsx

```
import React, { useContext } from 'react';
import { useParams } from 'react-router-dom';
import { BlogContext } from '../BlogContext/BlogPosts';

export default function BlogDetails() {
  const { index } = useParams();
  const { posts } = useContext(BlogContext);

  const post = posts[index];
  if (!post) return <p>Blog not found!</p>;

  return (
    <div>
      <h2>{post.title}</h2>
      <p>{post.content}</p>
    </div>
  );
}
```

Advantages of Context + Routing

✅Centralized data (easy to manage)
✅Cleaner and modular structure
✅No prop drilling
✅Data persistence using local storage
✅Simple and scalable navigation

Exercise

Build This Mini Project:

Goal: Simple Blog App using Context + Routing

Requirements:

1. Create 4 Components: `Home`, `Blogs`, `AddBlog`, `BlogDetails`.
2. Use `Context API` to manage posts globally.
3. Use `localStorage` to persist data.
4. Use `react-router-dom` for navigation.
5. Each blog post should show detailed content when clicked.
6. Add styling to make it look clean and minimal.

✅**Bonus:** Add a delete button for each blog and a confirmation dialog.

🚢Learning Outcome

By the end of this task, you'll fully understand: - Routing in React (`react-router-dom`) - Context API for global state - Using Local Storage for persistence - Component communication using Context - Managing CRUD operations in React