

Enhancing Navigation and Data Gathering with Layouts and Templates in Next.js

Data Gathering with Layouts:

To navigate across different links and gather data from users, we can employ input elements within a layout component, leveraging the `useState` component from React. First, we import `useState` from React. Then, within the layout component, we invoke `useState`. Before the `NavLink` elements, we add a `div` tag containing an input element for users to input data. This structure ensures that the layout retains all common elements.

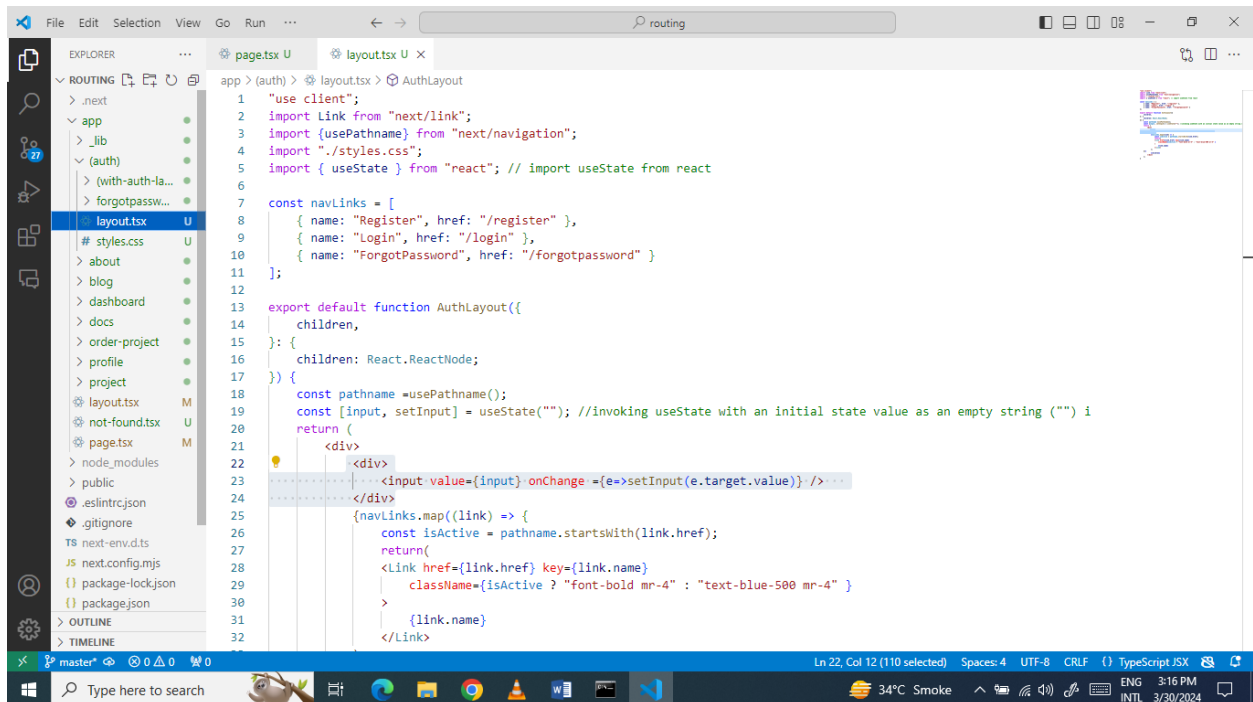
However, in certain scenarios where we require distinct instances of children, templates offer a solution. By using templates instead of the layout component, we can address such situations effectively.

Data Gathering with Templates:

Templates are similar to layouts in that they wrap each child layout or page. However, with templates, when a user navigates between routes that share a template, a new instance of the component is mounted. This means that DOM (Document Object Model) elements are recreated, state is not preserved, and effects are resynchronized. A template can be defined by exporting a default React component from a `template.js` or `template.tsx` file. Similar to layouts, templates also should accept a `children` prop, which will render the nested segments in the route. This structure allows for modularization and reusability across different parts of the application, ensuring a consistent user experience while maintaining flexibility in page composition.

"Data Gathering with Layouts in Next.js : Step-by-Step Guide with Screenshots"

Go to the (auth) folder and open layout.tsx file, make changes as commented and highlights

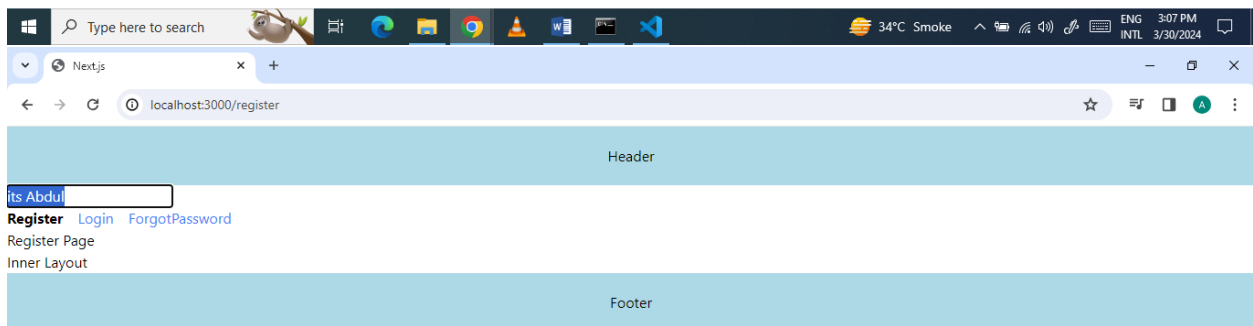
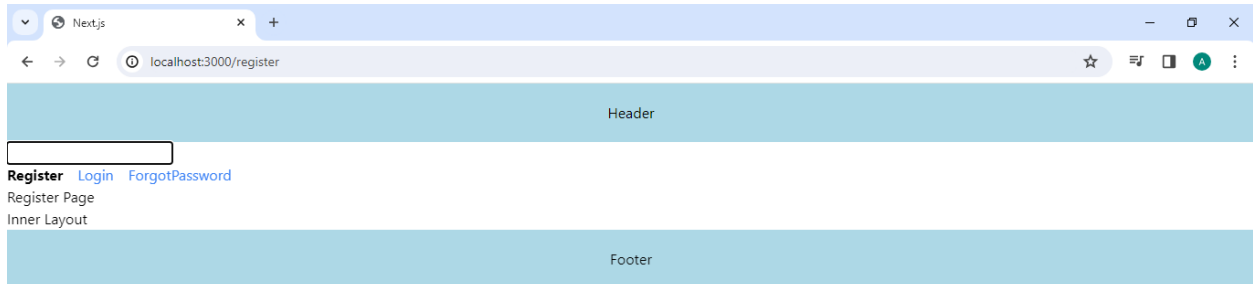


```
1 "use client";
2 import Link from "next/link";
3 import {usePathname} from "next/navigation";
4 import "../styles.css";
5 import { useState } from "react"; // import useState from react
6
7 const navLinks = [
8   { name: "Register", href: "/register" },
9   { name: "Login", href: "/login" },
10  { name: "ForgotPassword", href: "/forgotpassword" }
11 ];
12
13 export default function AuthLayout({
14   children,
15 }): {
16   children: React.ReactNode;
17 } {
18   const pathname = usePathname();
19   const [input, setInput] = useState(""); //invoking useState with an initial state value as an empty string ("")
20   return (
21     <div>
22       <input value={input} onChange={e=>setInput(e.target.value)} />
23     </div>
24     {navLinks.map((link) => {
25       const isActive = pathname.startsWith(link.href);
26       return(
27         <Link href={link.href} key={link.name}
28           className={isActive ? "font-bold mr-4" : "text-blue-500 mr-4" }
29         >{link.name}</Link>
30       )
31     })}
32   )
33 }
```

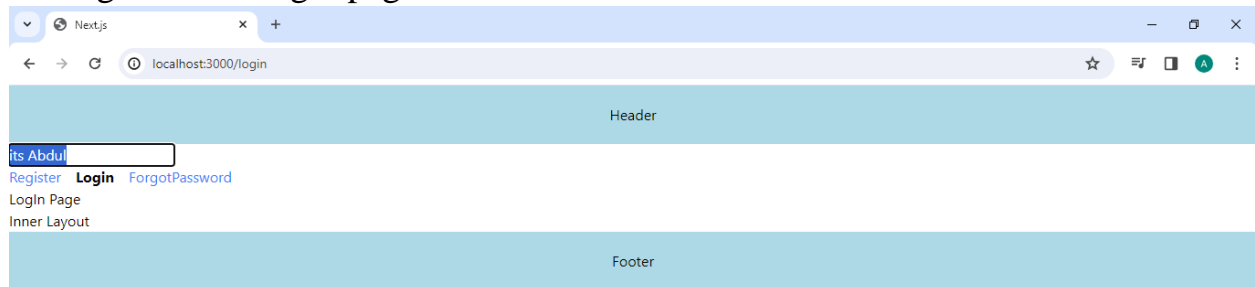
Go to the browser and navigate to localhost:3000.



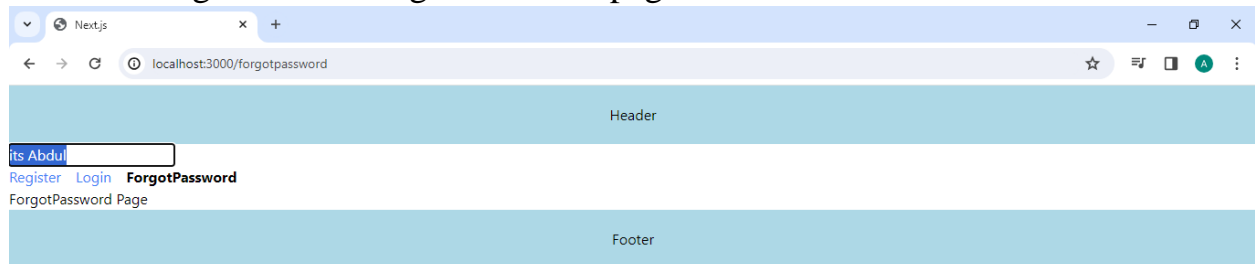
Navigate to localhost:3000/register and observe the input fields. Add some input text, then navigate to the login or forgot password page. The same text will be available there.



"Navigate to the login page. The same text will be available."

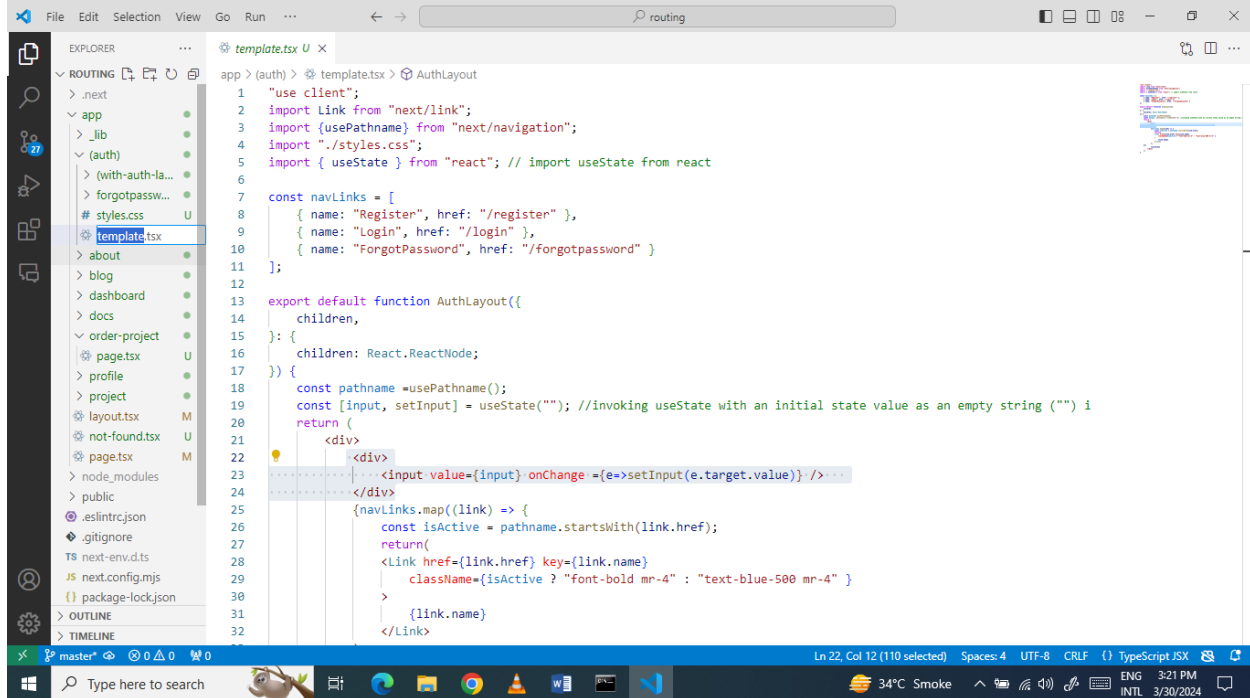


"Navigate to the ForgotPassword page. The same text will be available."



"Data Gathering with Templates in Next.js : Step-by-Step Guide with Screenshots"

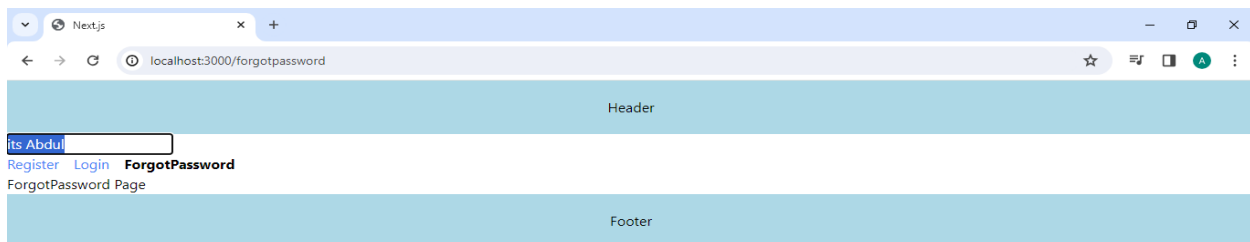
"For this, go to the (auth) folder and rename the layout.tsx file to template.tsx."

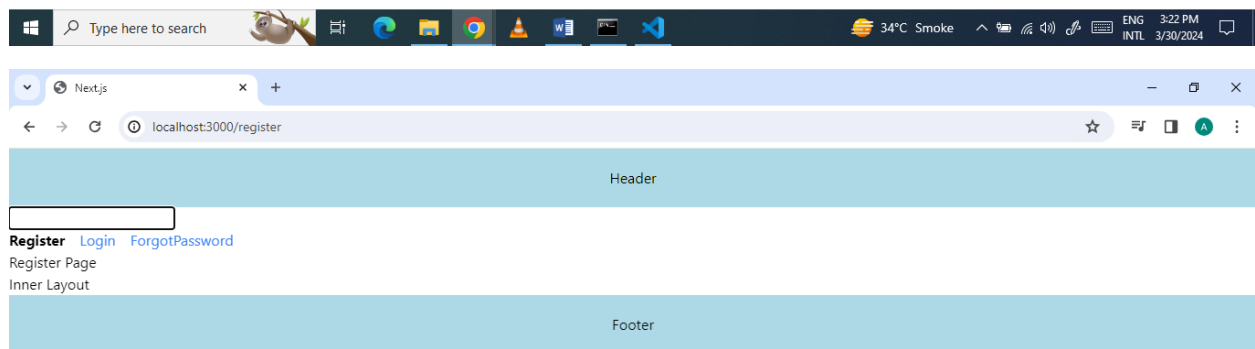
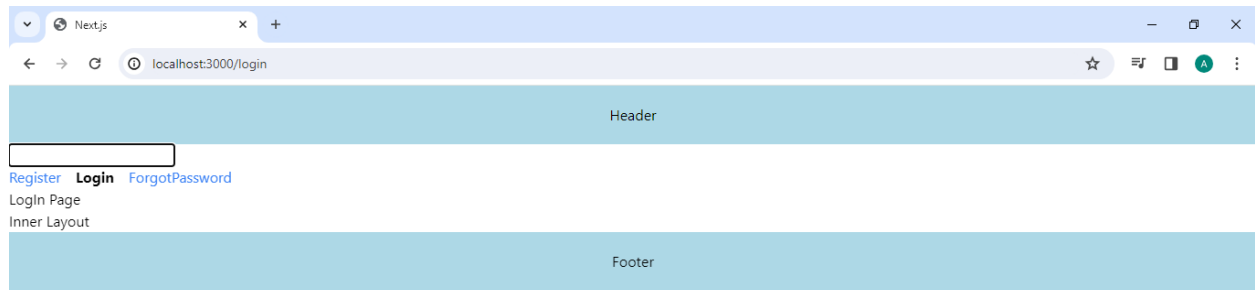


The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the project structure, with the 'app' folder expanded, and the 'auth' folder selected. The 'template.tsx' file is highlighted in the file explorer. The main editor window displays the code for 'template.tsx'. The code includes imports for 'use client', 'Link', 'usePathname', 'styles.css', and 'useState'. It defines a 'navLinks' array with links for 'Register', 'Login', and 'ForgotPassword'. The 'AuthLayout' function is defined, which takes 'children' as a prop and returns a JSX element. The JSX element includes an input field for the 'ForgotPassword' page and a navigation bar with links for 'Register', 'Login', and 'ForgotPassword'. The 'className' for the links is determined by the 'pathname' and the 'link.href'.

```
1 "use client";
2 import Link from "next/link";
3 import {usePathname} from "next/navigation";
4 import "../styles.css";
5 import { useState } from "react"; // import useState from react
6
7 const navLinks = [
8   { name: "Register", href: "/register" },
9   { name: "Login", href: "/login" },
10  { name: "ForgotPassword", href: "/forgotpassword" }
11 ];
12
13 export default function AuthLayout({
14   children,
15 }): {
16   children: React.ReactNode;
17 } {
18   const pathname = usePathname();
19   const [input, setInput] = useState(""); // invoking useState with an initial state value as an empty string ("")
20   return (
21     <div>
22       <input value={input} onChange={e=>setInput(e.target.value)} />
23     </div>
24     <div>
25       {navLinks.map((link) => {
26         const isActive = pathname.startsWith(link.href);
27         return(
28           <Link href={link.href} key={link.name}
29             className={isActive ? "font-bold mr-4" : "text-blue-500 mr-4"}
30           >{link.name}</Link>
31         )
32       })}
33     </div>
34   );
35 }
```

"Go to the browser and write something in the input field. Then, navigate to another page. You will notice that the state is no longer preserved. This illustrates the significance of template files in the new app router."





It is possible that both the Layout file and the Template file can be utilized. In this scenario initially, the layout is rendered first, and subsequently, the layout's children are replaced with the component exported by the template file.