

Active Link in Next.js

Active Link in Next.js:

An active link in Next.js is a link on a webpage that tells you which page you are currently on. It's like a highlighted or differently styled link that shows you where you are in the website's navigation. This feature helps users understand their current location within the website and makes navigation easier.

In Next.js, "active links" typically refer to the feature provided by the Link component from the next/link module. The purpose of active links is to enhance user experience by indicating which page the user is currently viewing or navigating to.

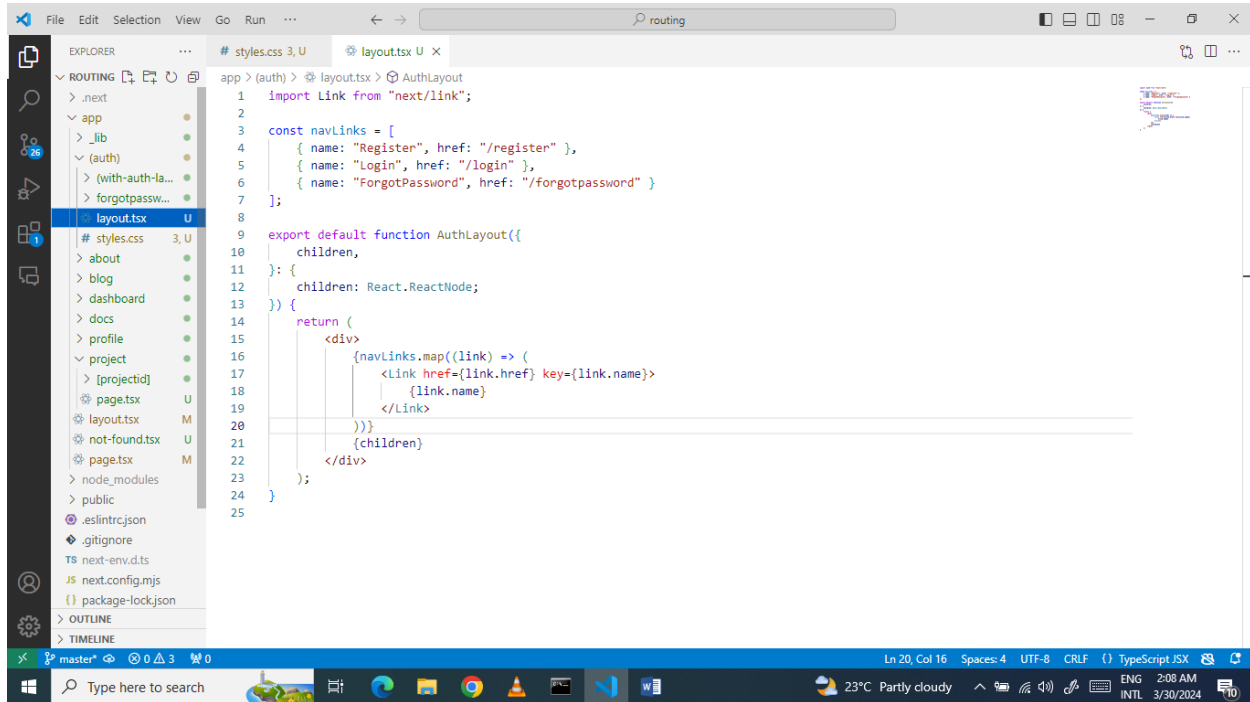
When you navigate between pages using Next.js, the browser typically reloads the entire page. However, Next.js provides the Link component to handle client-side navigation without a full page reload, resulting in faster navigation and a smoother user experience.

The Link component allows you to create links between pages in your Next.js application. When a user clicks on a link created with the Link component, Next.js intercepts the click event and loads the linked page's content dynamically without a full page reload. This behavior is similar to single-page application (SPA) frameworks like React Router.

The "active link" feature refers to the ability of the Link component to apply a specific styling or class to indicate that the link corresponds to the current active page. This helps users visually identify their current location within the application's navigation structure

"Active Link in Next.js : Step-by-Step Guide with Screenshots"

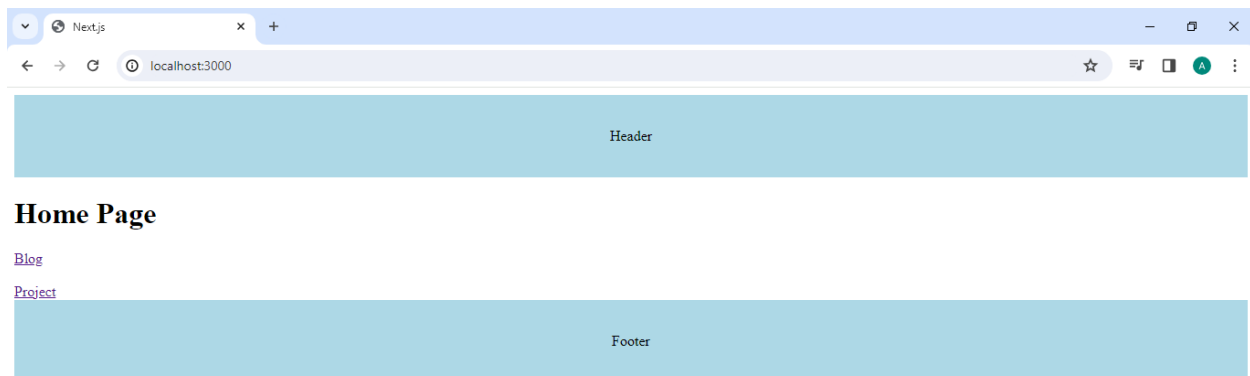
"Inside the (auth) folder, create a layout.tsx file."



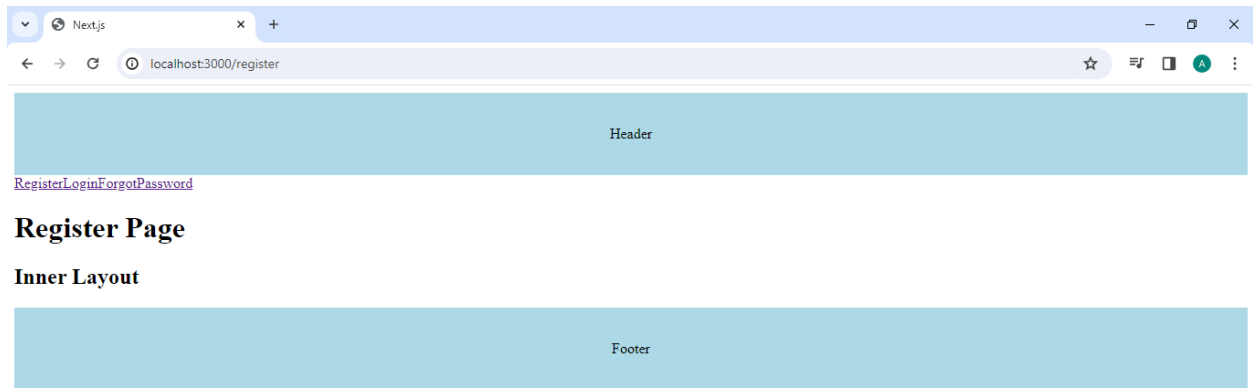
The screenshot shows a Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a folder named 'auth' containing a 'layout.tsx' file. The code editor shows the content of 'layout.tsx' with the following code:

```
1 import Link from "next/link";
2
3 const navLinks = [
4   { name: "Register", href: "/register" },
5   { name: "Login", href: "/login" },
6   { name: "ForgotPassword", href: "/forgotpassword" }
7 ];
8
9 export default function AuthLayout({
10   children,
11 }) {
12   children: React.ReactNode;
13 } {
14   return (
15     <div>
16       {navLinks.map((link) => (
17         <Link href={link.href} key={link.name}>
18           {link.name}
19         </Link>
20       ))}
21       {children}
22     </div>
23   );
24 }
25
```

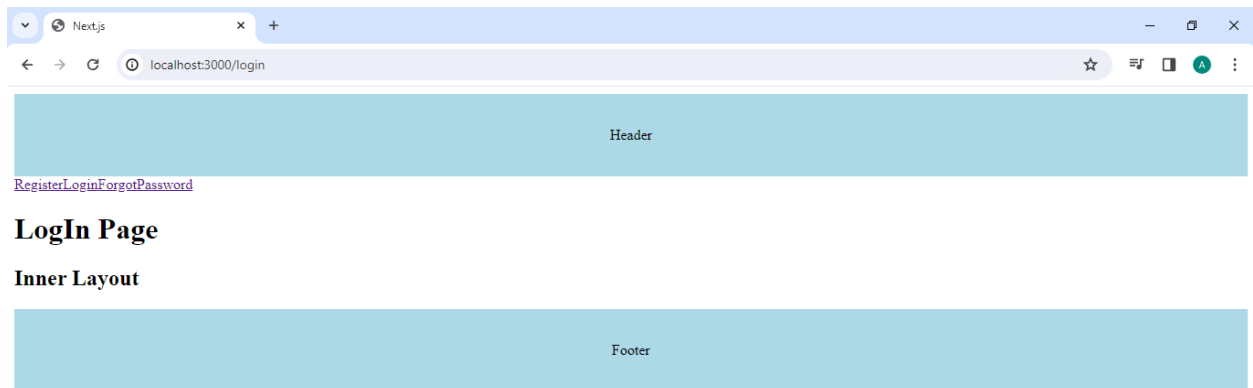
Now, run the command **npm run dev**, and then the home page will be displayed.

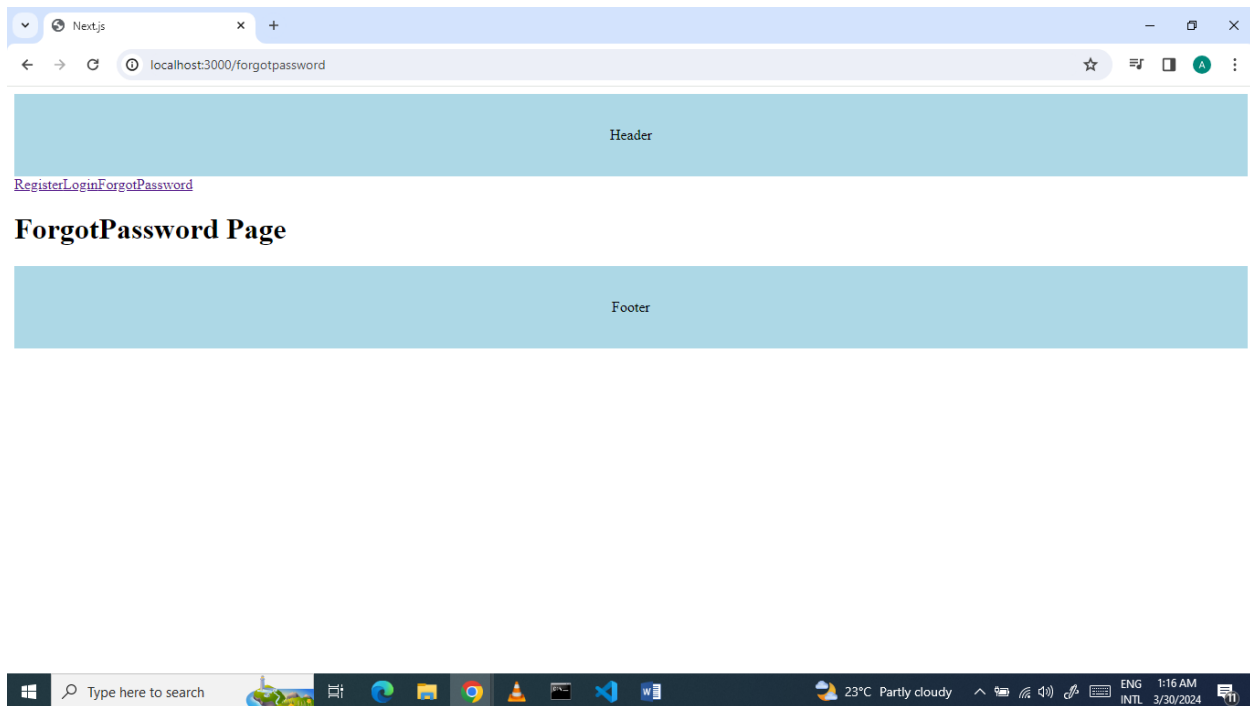


"Navigate to localhost:3000/register. After doing so, the links for registration, login, and password recovery will become available."

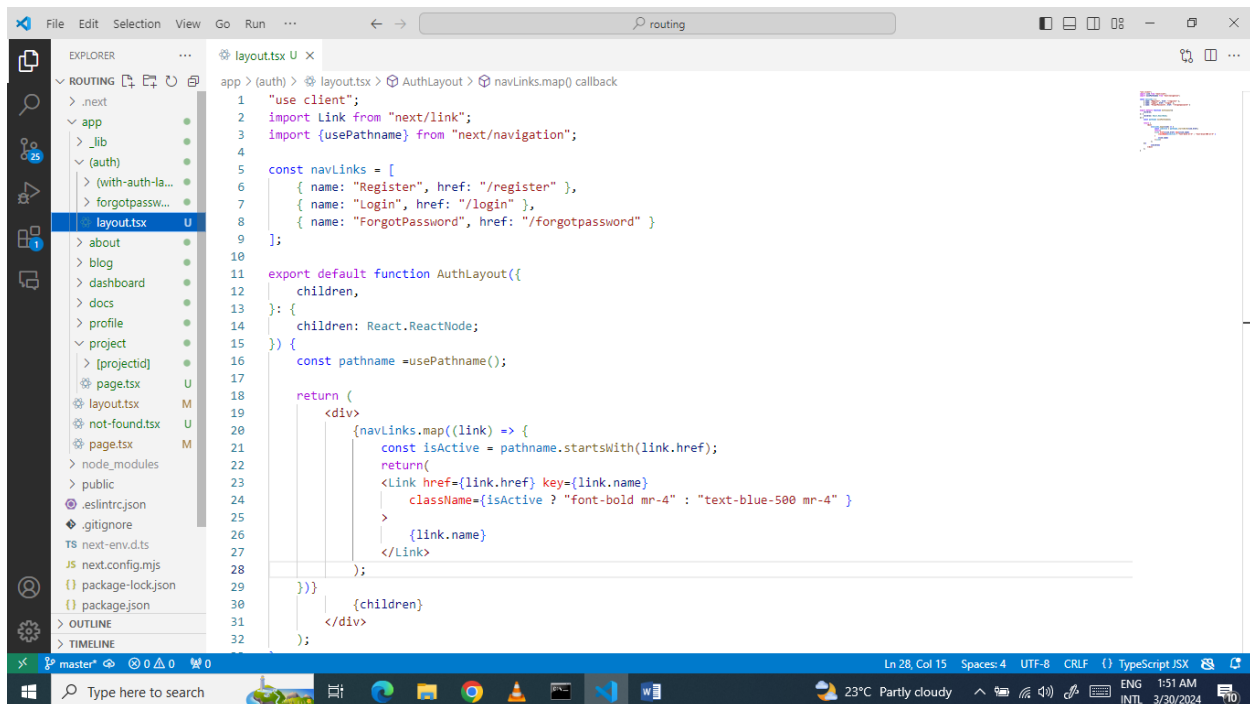


"Now navigate through them one by one and observe the active links."

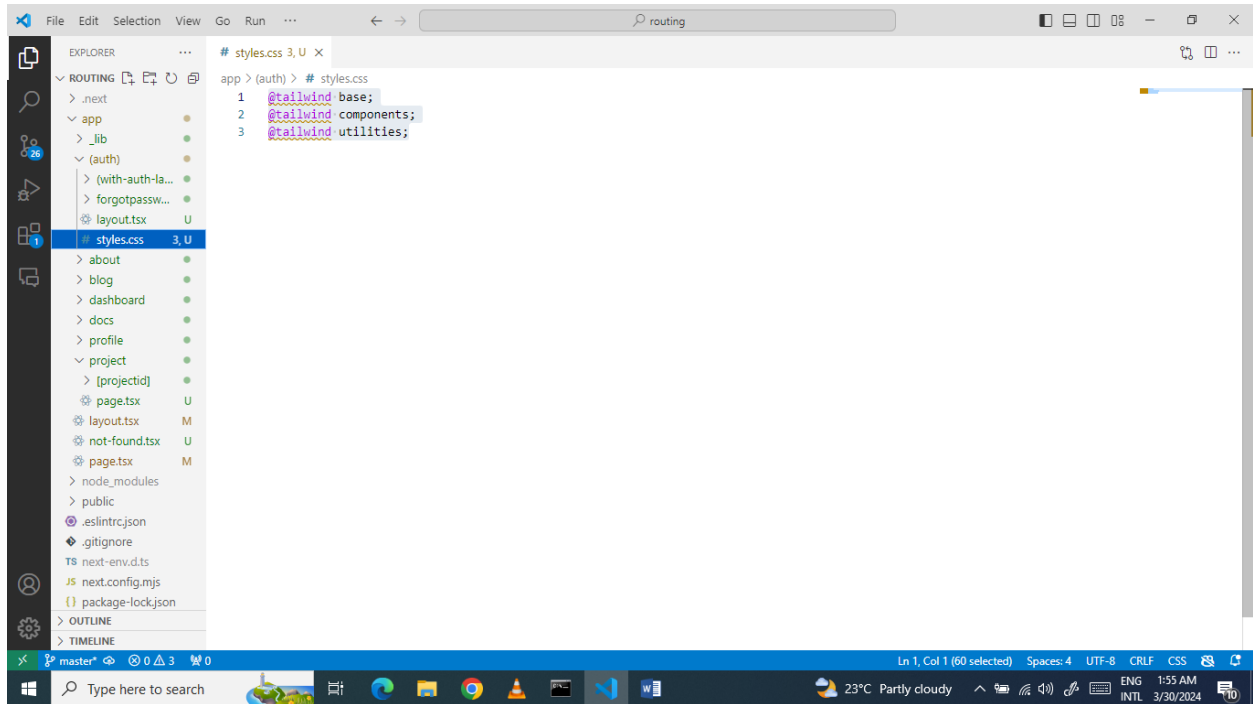




"Now, style the active links by making changes to the layout.tsx file as shown below:"

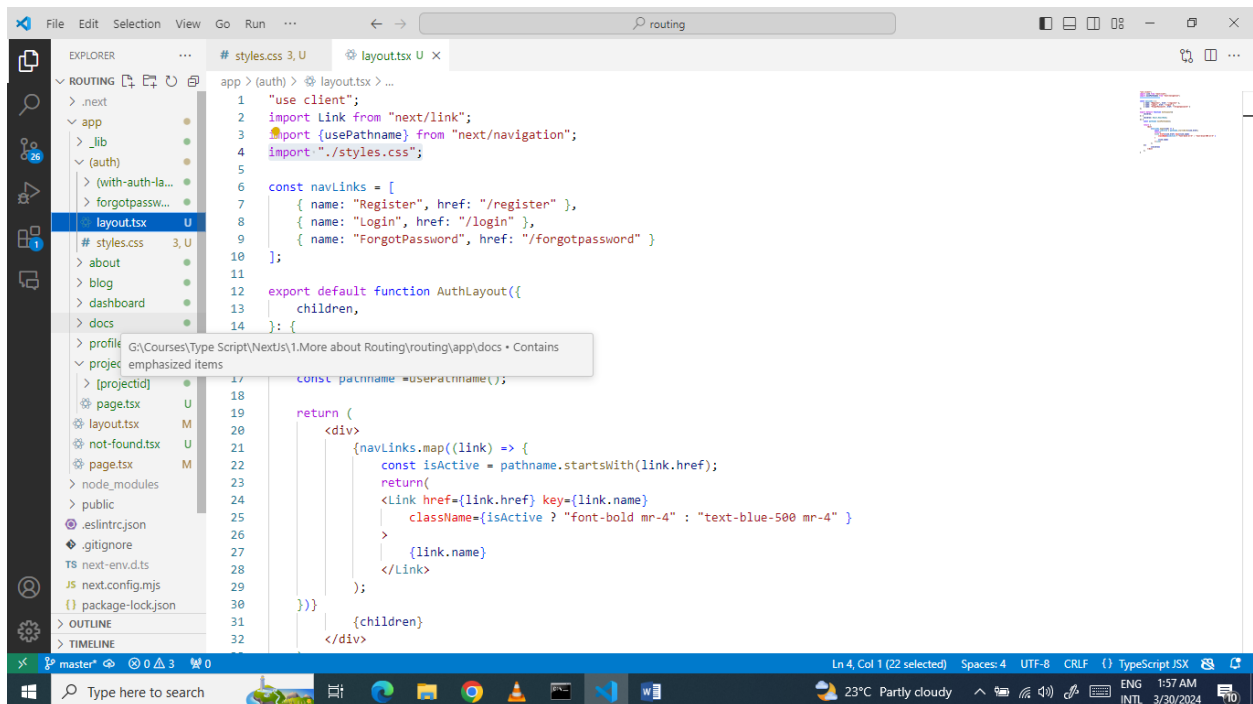


"Create a styles.css file inside the (auth) folder and edit it."



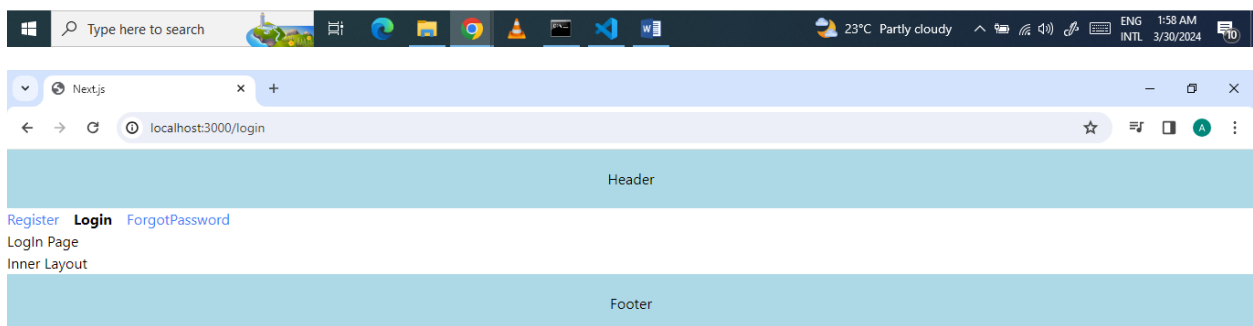
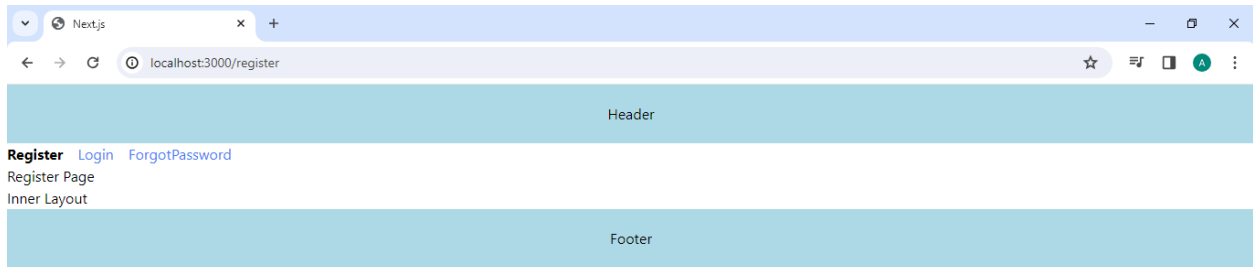
The screenshot shows the Visual Studio Code interface. The Explorer panel on the left displays the file structure of a project, with the 'auth' folder expanded. Inside 'auth', there is a 'styles.css' file. The main editor window shows the content of 'styles.css', which contains three lines of Tailwind CSS directives: `@tailwind base;`, `@tailwind components;`, and `@tailwind utilities;`. The status bar at the bottom indicates the file is 'Ln 1, Col 1 (60 selected)' and is using 'Spaces: 4', 'UTF-8', 'CRLF', and 'CSS' encoding.

"Import the styles.css file inside the layout.tsx file of the (auth) folder."



The screenshot shows the Visual Studio Code interface with the 'layout.tsx' file open in the main editor. The Explorer panel on the left shows the 'auth' folder expanded, with 'layout.tsx' selected. The main editor displays the code for 'layout.tsx'. Lines 1-4 show imports: `"use client";`, `import Link from "next/link";`, `import {usePathname} from "next/navigation";`, and `import "../styles.css";`. Lines 6-10 define a `navLinks` array with three objects. Lines 12-14 define an `AuthLayout` function. Lines 17-32 show the `render` function, which uses `usePathname()` and `navLinks` to render a list of links. The status bar at the bottom indicates the file is 'Ln 4, Col 1 (22 selected)' and is using 'Spaces: 4', 'UTF-8', 'CRLF', and 'TypeScript JSX' encoding.

"In the browser, the current link is displayed in bold black, while the remaining links are shown in blue."



Active links are super important in Next.js apps because they help users know where they are on the website and make it easier to move around. With active links, you can click on a link and smoothly move to another page without the whole page reloading. This makes browsing faster and smoother. By styling active links nicely, you not only help users find their way but also make your website look better. So, by using and styling active links well, developers can make their Next.js projects more user-friendly and enjoyable for everyone.