Enhancing User Experience with Loading States in Next.js

In Next.js, several special files play key roles in managing the application's behavior and user experience:

page.tsx:

This file represents a specific page in the application and is responsible for rendering its content.

layout.tsx:

Layout files define the overall structure of a page, including common elements like headers, footers, and navigation menus. These elements remain consistent across different pages, ensuring a cohesive user experience.

template.tsx:

Templates serve as reusable components that wrap each child layout or page. They provide a consistent structure for pages sharing similar designs or functionalities.

notfound.tsx:

This file handles rendering when a requested page is not found or does not exist.

Additionally, Next.js introduces another special file:

loading.tsx:

This file enables the creation of loading states that display to users while content for a specific route segment is loading. The loading state appears immediately upon navigation, reassuring users that the application is responsive and actively fetching content.

Benefits of loading.tsx:

Displaying the loading state immediately:

When you go to a new page on a website, if there's a loading animation or message that pops up right away, it lets you know that the website is working on getting the new page ready. This makes you feel like the website is quick to respond to your actions and makes you feel like you're not waiting as long for the new page to load.

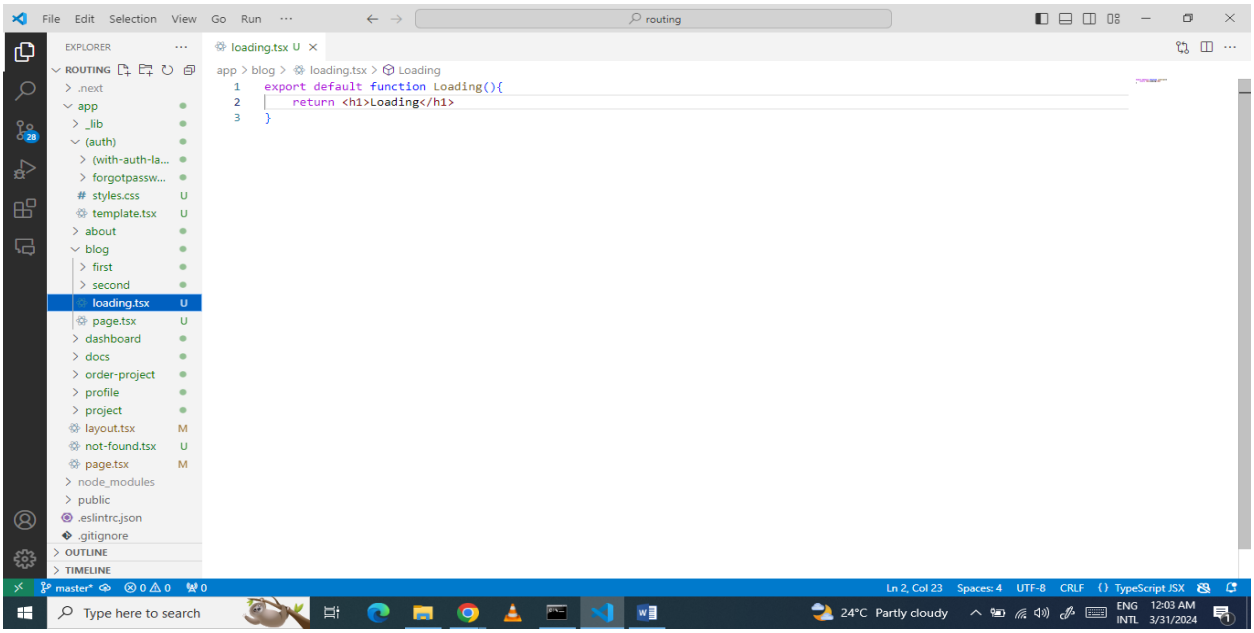
Shared layouts in Next.js:

Next.js lets you create layouts that stay the same across different pages of your website. Even if a new page is loading, parts of the website that are always there, like the menu or sidebar, can still be used. So, you can keep navigating around or using those parts of the website while waiting for the new page to fully load.

" To create a loading state in Next.js : Step-by-Step Guide with Screenshots"

To create a loading state, simply add or create a loading.tsx file in the designated folder.

For instance, let's consider implementing a loading state for the blog folder. Create a loading.tsx file inside the blog folder and add a React component.



Go to the browser and navigate to http://localhost:3000. Click on the blog link and observe that a loading indicator will display for some time, then it will be replaced with the blog text.