

Parallel Intercepting Routes in Next.js

Parallel Intercepting Routes:

In Next.js, parallel intercepting routes refer to a routing pattern where multiple routes can be active simultaneously and can intercept each other's navigation events. This allows for more complex and dynamic user interfaces where different parts of the application can respond to navigation changes independently.

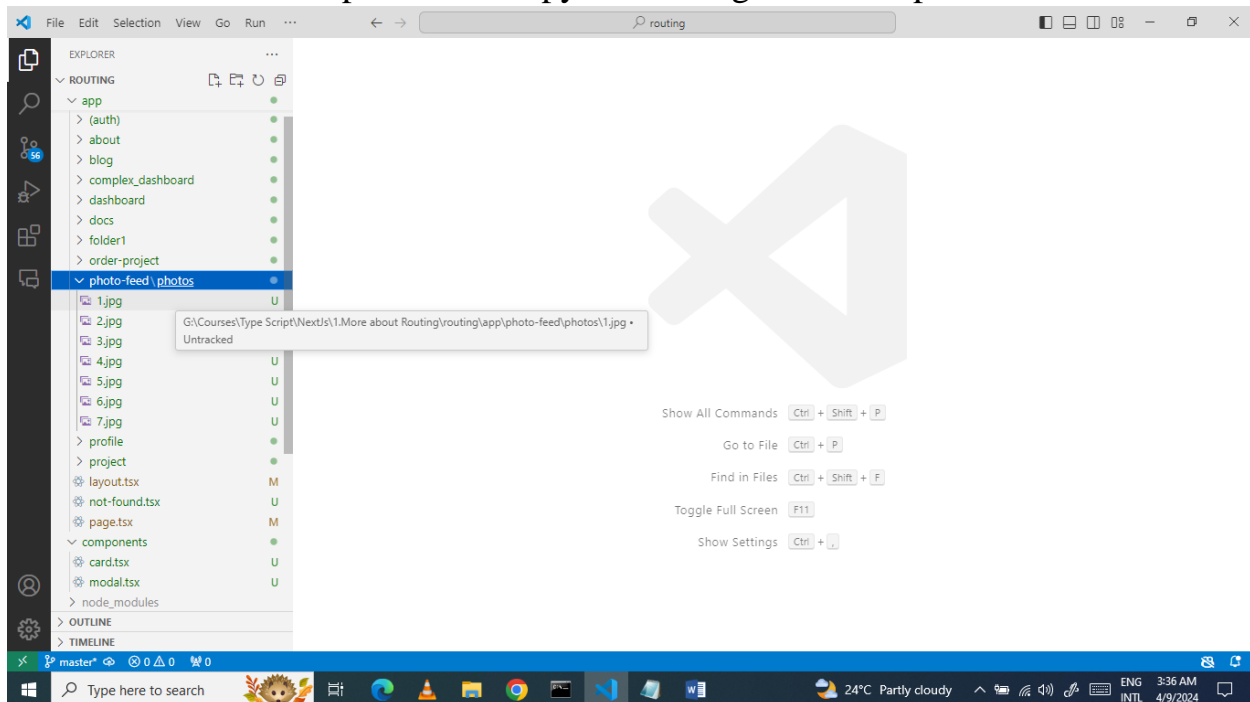
The purpose of parallel intercepting routes in Next.js is to provide developers with greater flexibility and control over the navigation flow within their applications. This pattern enables the creation of features such as modals, overlays, or sidebars that can overlay on top of the main content without disrupting the underlying page navigation.

By intercepting routes in parallel, developers can ensure that certain components or UI elements remain persistent across different pages or views, providing a smoother and more seamless user experience. For example, a modal dialog for displaying additional information or capturing user input can be triggered without navigating away from the current page, enhancing the interactivity and usability of the application.

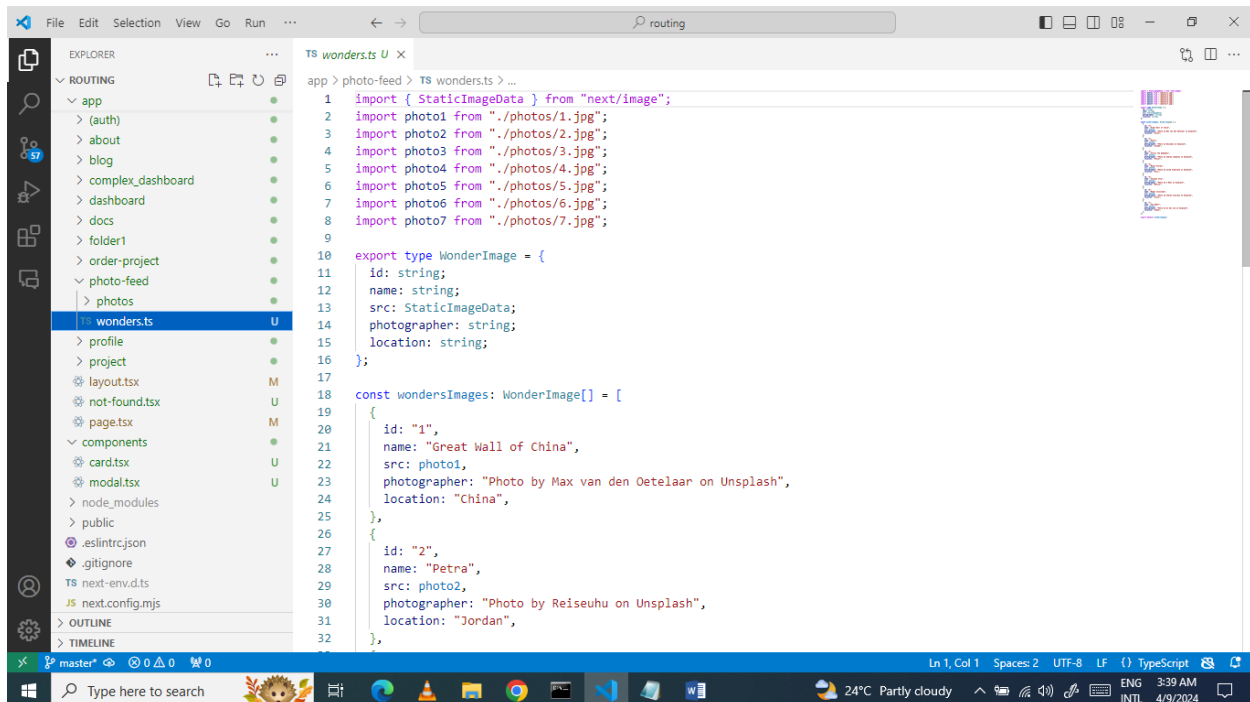
Overall, parallel intercepting routes in Next.js offer a powerful mechanism for building rich and interactive user interfaces while maintaining the integrity of the application's navigation structure.

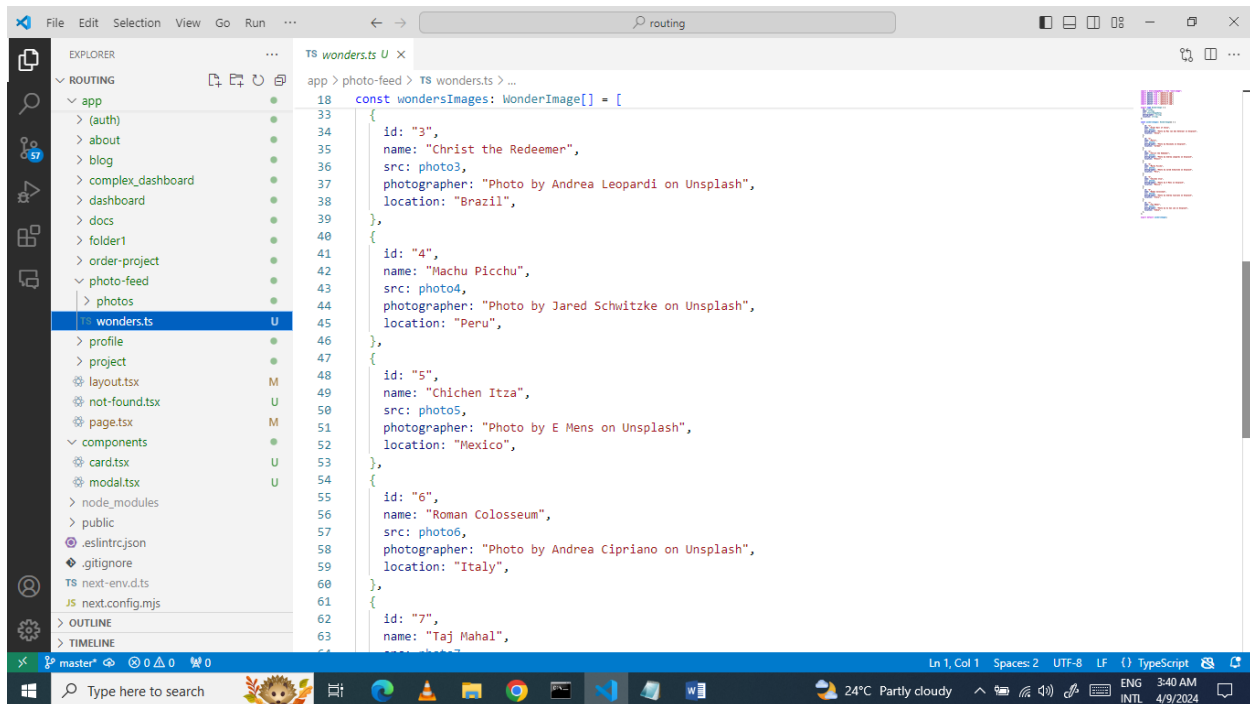
A modal is an ideal use case for parallel intercepting routes in Next.js due to its ability to overlay content seamlessly without disrupting the underlying page navigation. By intercepting routes in parallel, modals can persistently display additional information or interactive elements while maintaining context, thereby enhancing the user experience by providing a non-intrusive way to present dynamic content and allowing for flexible interaction patterns within the application interface.

"Create a photo-feed folder in the app folder. Inside the photo-feed folder, create another folder named 'photo' and copy some images into the photo folder."

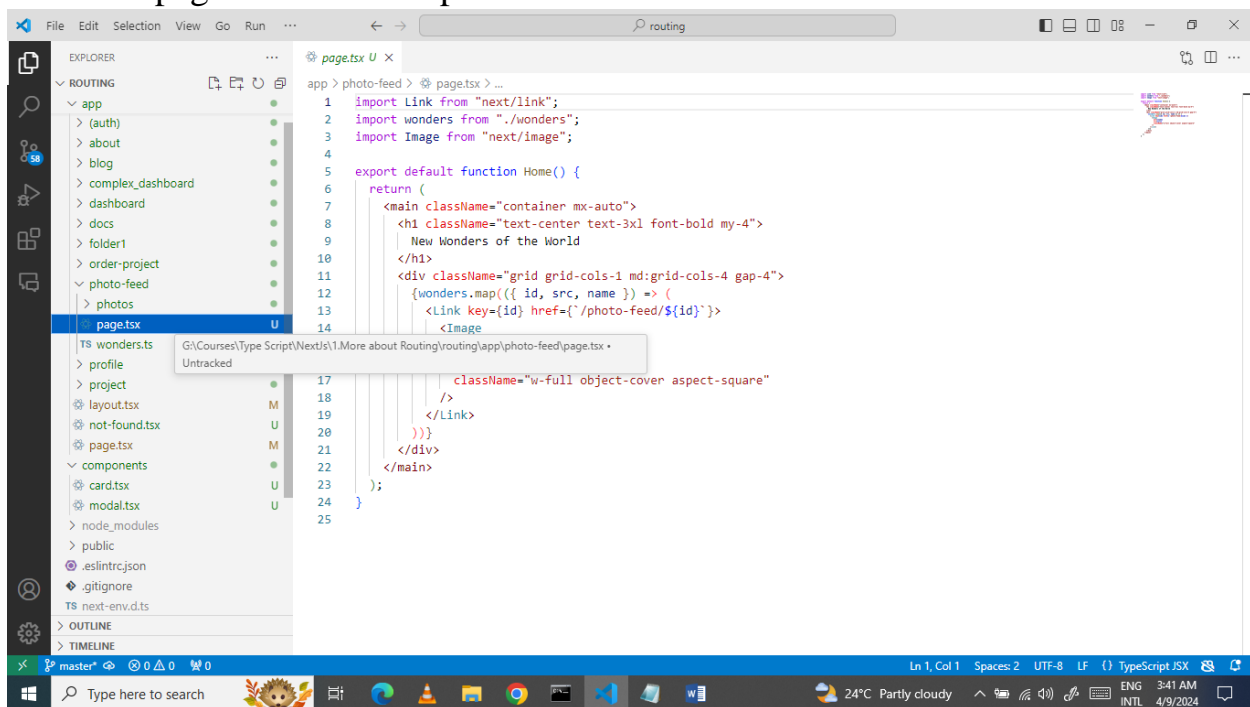


"Create a wonder.ts file inside the photo-feed folder and code as shown in the screenshots below."

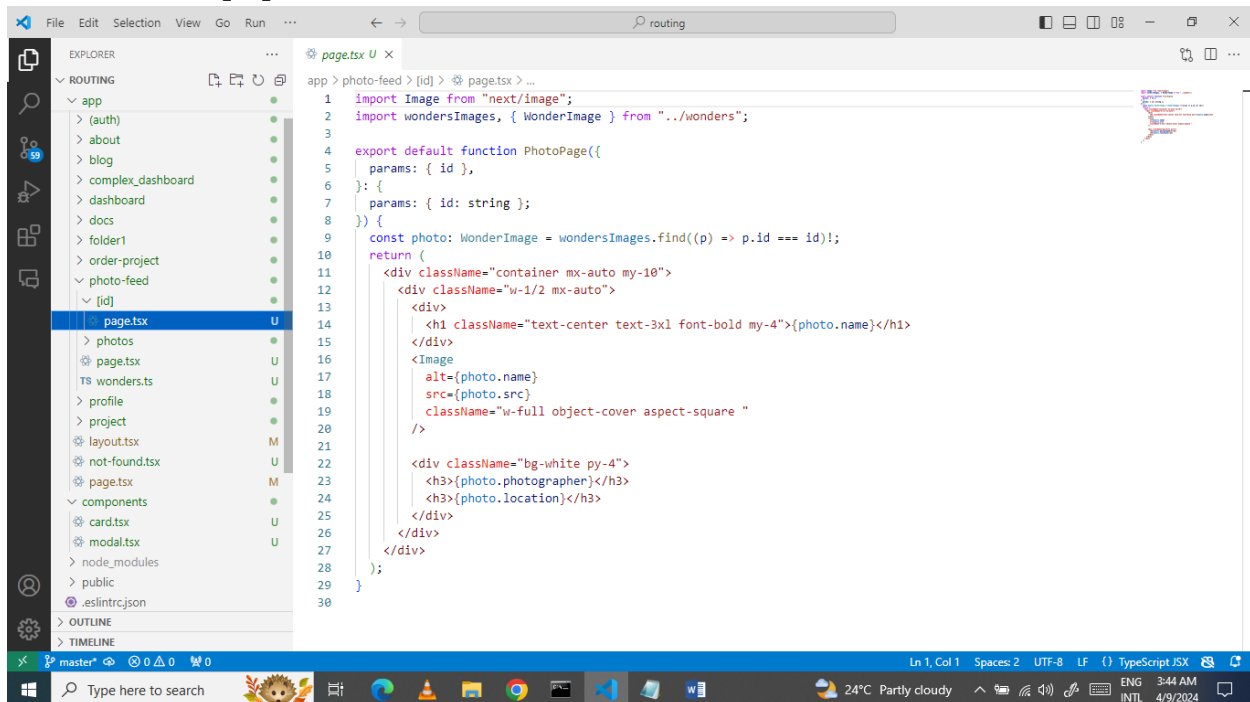




"Create a page.tsx file in the photo feed folder."



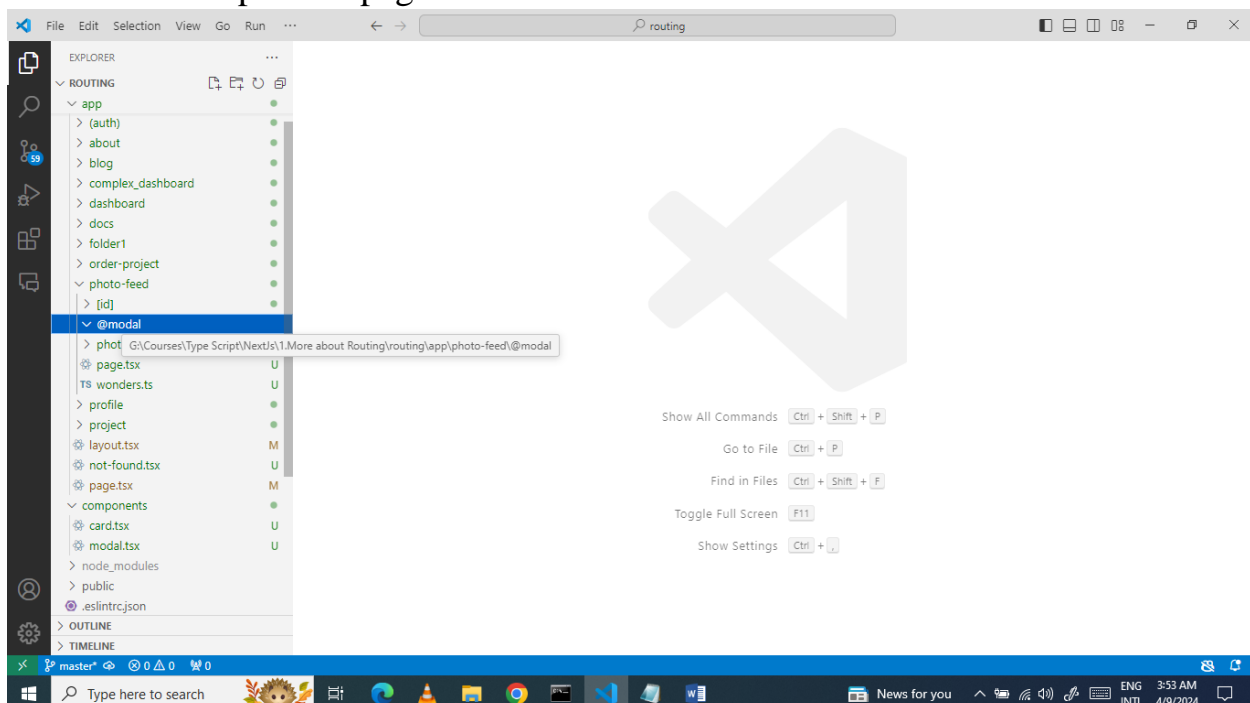
"Create a photo details dynamic route in the photo feed folder and create a page.tsx file inside the [id] folder."



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the project structure. The 'photo-feed' folder is expanded, showing a subfolder '[id]'. Inside '[id]', a 'page.tsx' file is selected. The main editor window shows the content of 'page.tsx'.

```
1 import Image from "next/image";
2 import wondersImages, { WonderImage } from "../wonders";
3
4 export default function PhotoPage({
5   params: { id },
6 }): {
7   params: { id: string };
8 } {
9   const photo: WonderImage = wondersImages.find((p) => p.id === id!);
10  return (
11    <div className="container mx-auto my-10">
12      <div className="w-1/2 mx-auto">
13        <div>
14          <h1 className="text-center text-3xl font-bold my-4">{photo.name}</h1>
15        </div>
16        <Image
17          alt={photo.name}
18          src={photo.src}
19          className="w-full object-cover aspect-square "
20        />
21      <div className="bg-white py-4">
22        <h3>{photo.photographer}</h3>
23        <h3>{photo.location}</h3>
24      </div>
25    </div>
26  );
27 };
```

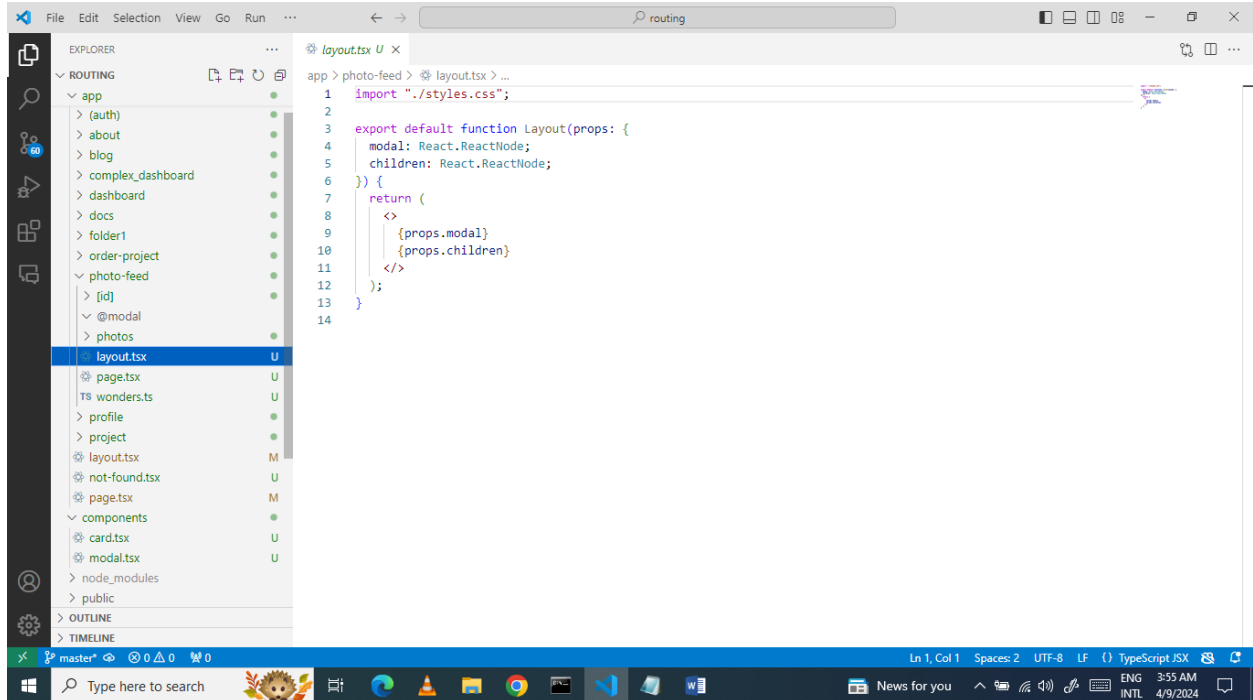
"Now we need to intercept the navigation from the feed to the details page since we want a modal that will render on top of the current feed. We need a parallel route. So, within the photo-feed folder, create a modal as a parallel route and include it in layout.tsx, along with the children slot. This way, we have the modal and children. Children correspond to page.tsx."



The screenshot shows the Visual Studio Code interface. The Explorer sidebar is expanded to show the 'photo-feed' folder. A new folder named '@modal' has been created inside 'photo-feed'. The main editor window is currently blank, displaying the VS Code logo and a list of keyboard shortcuts.

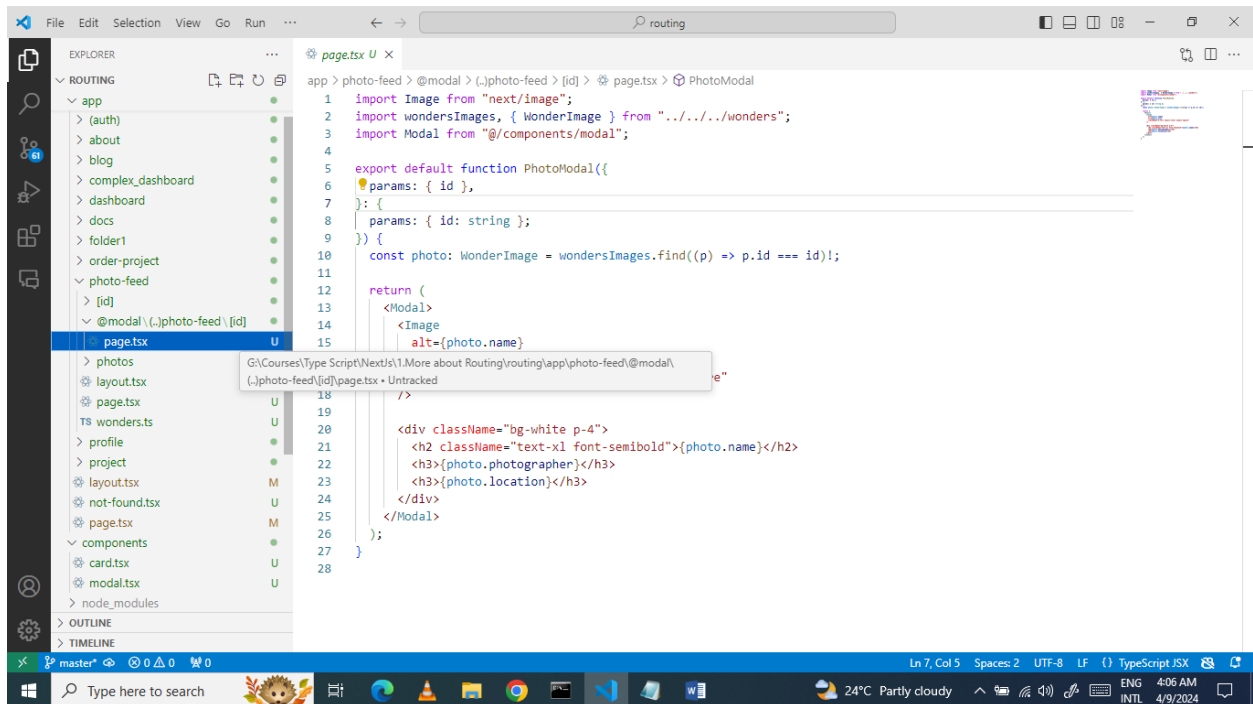
```
Show All Commands  Ctrl + Shift + P
Go to File         Ctrl + P
Find in Files       Ctrl + Shift + F
Toggle Full Screen  F11
Show Settings       Ctrl + ,
```

"Create a layout.tsx file inside the photo-feed folder."

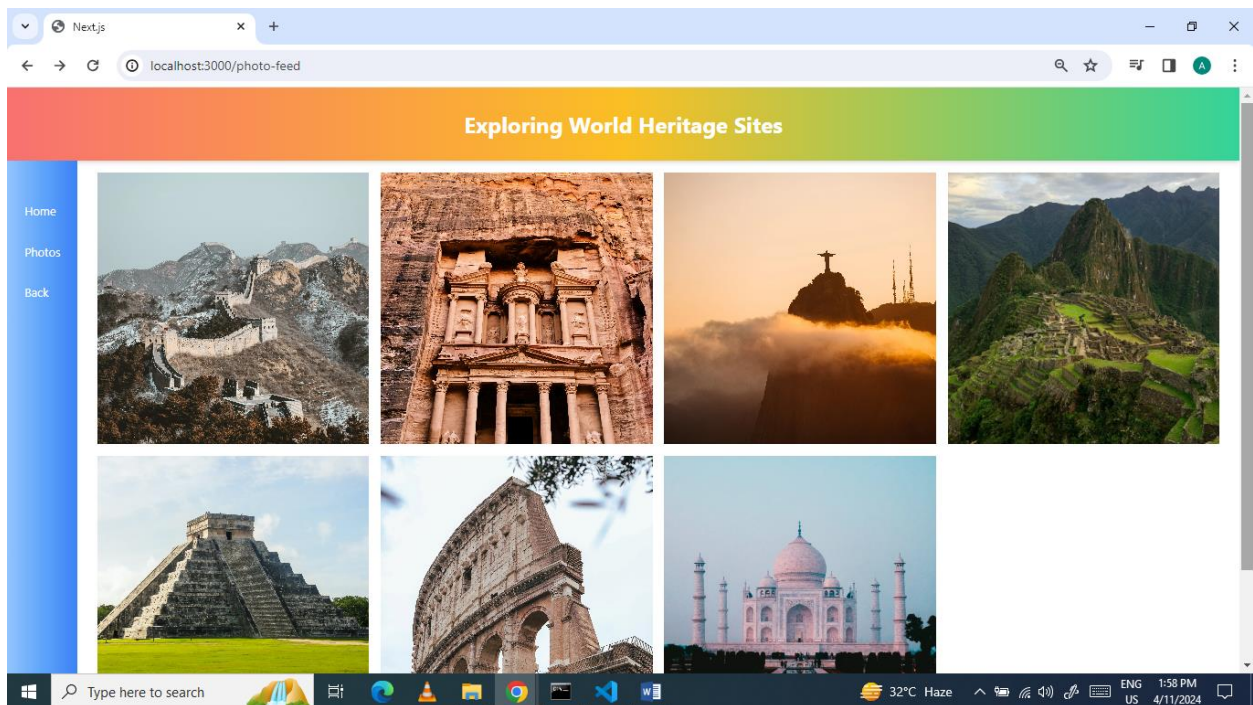


Create the photo-feed route named **((..)photo-feed)** inside the **@model** folder, which intercepts the photo feed route one segment above, also containing the ID folder. Within the ID folder, create a page that renders the photo details similar to the ID route from before, but this time placed inside a model. The photo name is positioned at the bottom instead of the top to demonstrate that we can have different user interfaces for the two routes: one for the original route and one for the intercepting route.

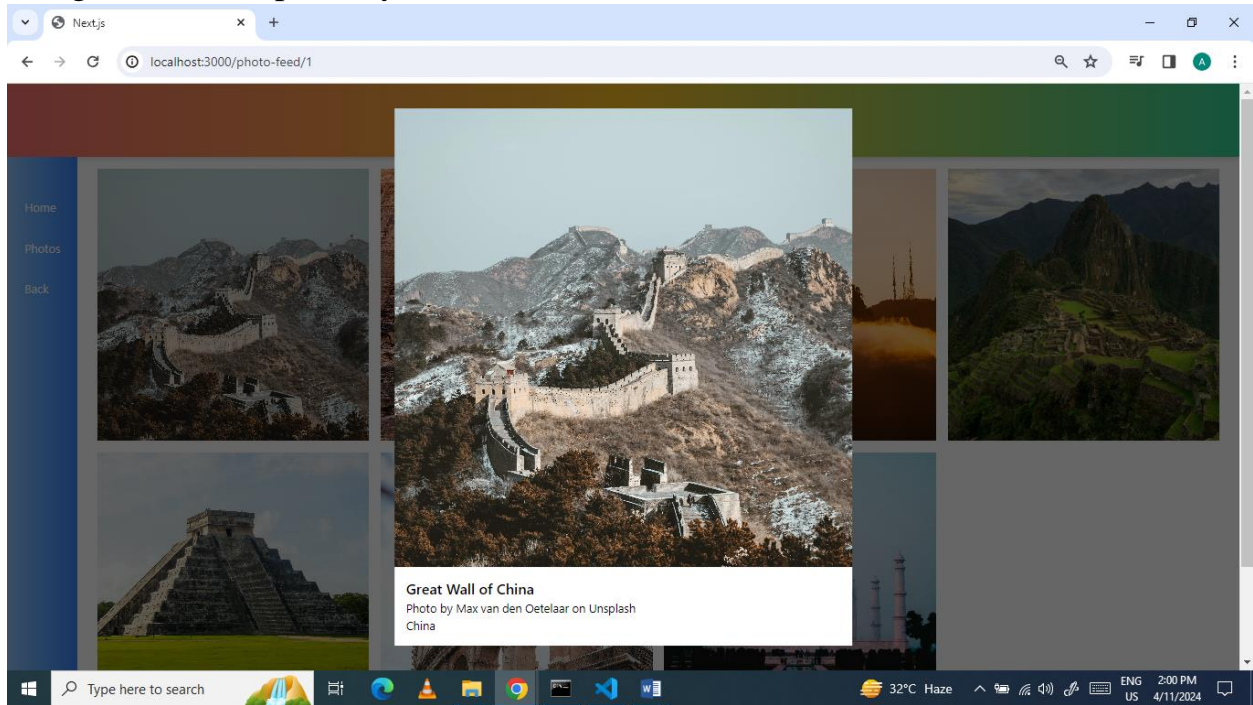
Inside the **((..)photo-feed)** folder, create an **[id]** folder, and within the **[id]** folder, create a **page.tsx** file as shown in the screenshots below.



"Now, go to the browser, navigate to <http://localhost:3000/photo-feed>"



Interfaces for the two routes differ—one for the original route and one for the intercepting route. With this setup, clicking on an image in the feed should trigger navigation interception by the model route.



If you reload the page, it will restore the original route.

