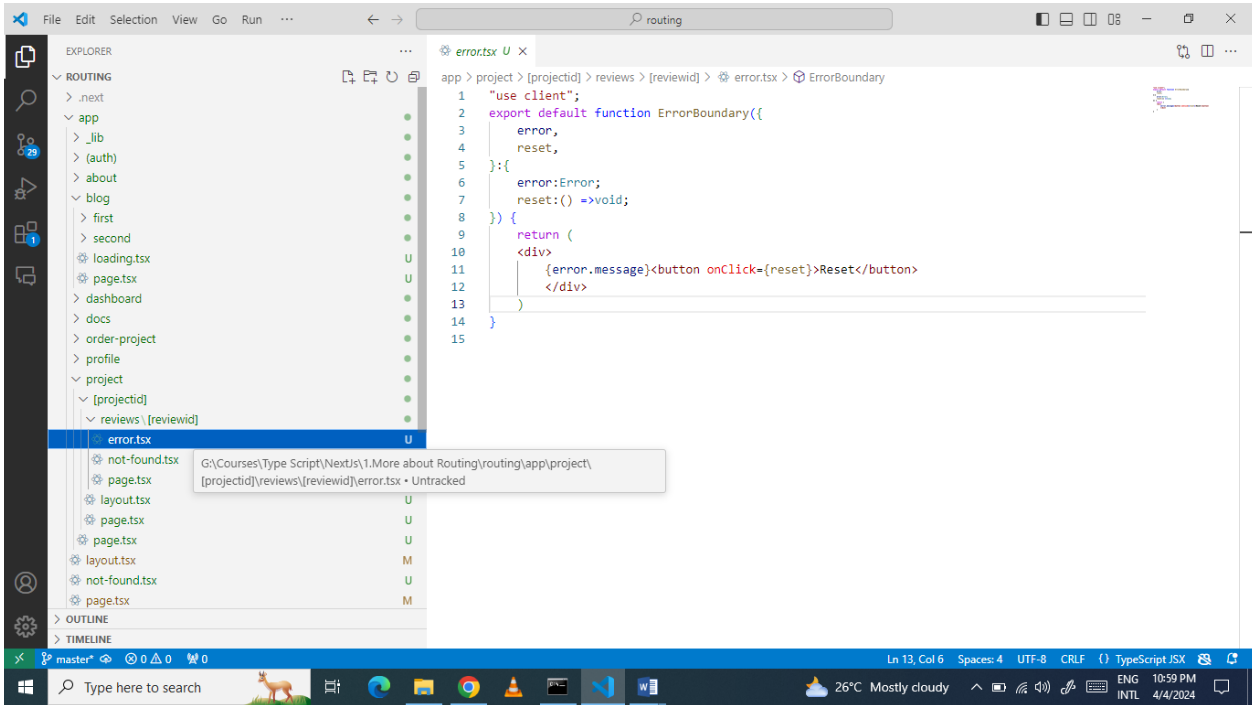
Handling Errors in Nested Routes in Next.js
Step-by-Step Guide with Screenshots

Note:

Continuing from the previous guide on Handling Errors in Next.js, this guide will discuss handling errors in nested routes in Next.js. In this guide, we will directly start practical implementation of handling errors in Nested Routes in Next.js.

It's important to know that errors move up to the closest error boundary. This means that an error file handles errors for all its nested parts. By putting error files at different levels in the route's nested folders, you can deal with errors more specifically.

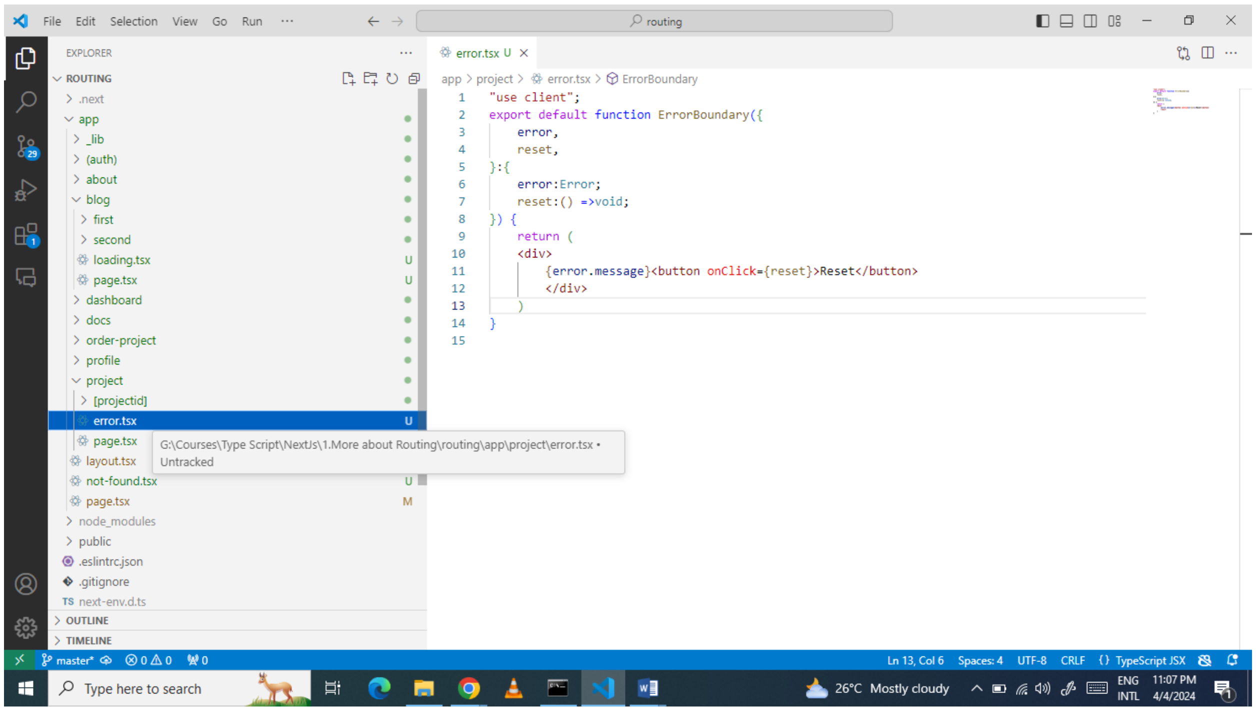
At present, our error file, error.tsx, resides within the review ID folder. Our route structure features a degree of nesting, starting with project at the highest level, followed by project ID, and then review ID. This nested arrangement of components directly affects the behavior of our error file, like the screen shots below.



```
G:\Courses\Type Script\NextJs\1.More about Routing\routing\app\project\[projectid]\reviews\[reviewid]\error.tsx • Untracked
```

```tsx
"use client";
export default function ErrorBoundary({
    error,
    reset,
}:{
    error:Error;
    reset:() =>void;
}) {
    return (
    <div>
        {error.message}<button onClick={reset}>Reset</button>
        </div>
    )
}
```

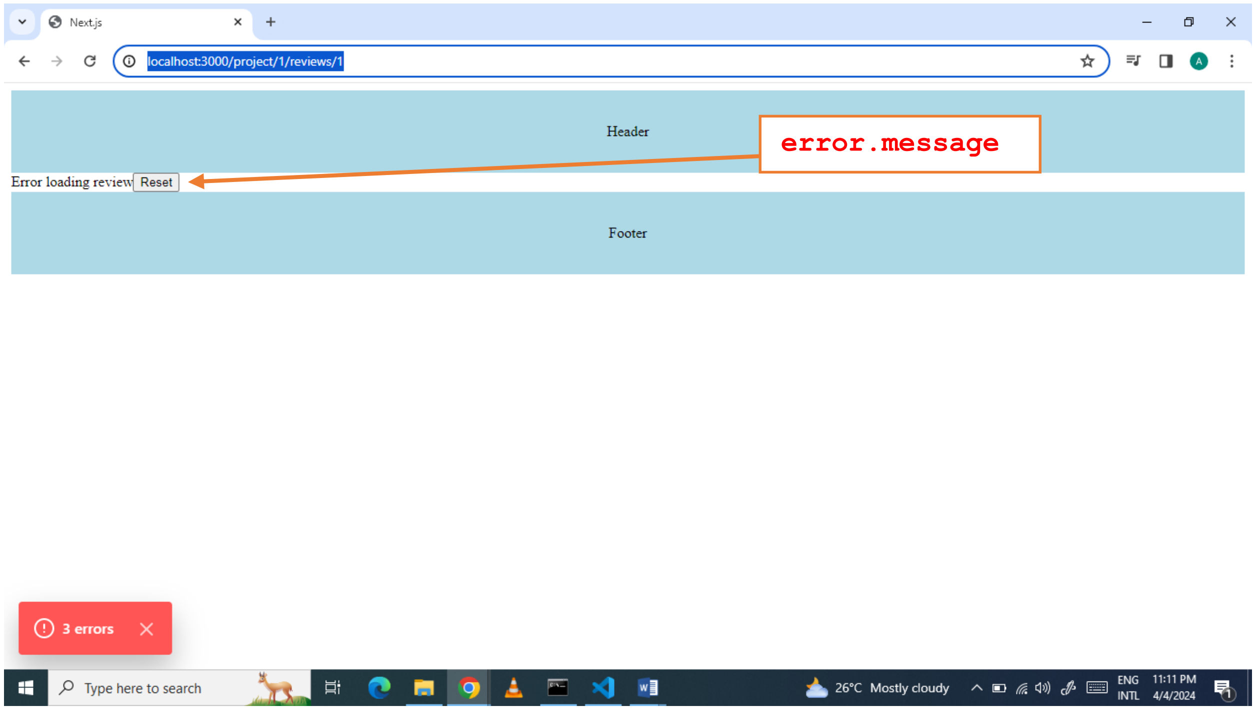
Move the error.tsx file from the review ID folder to the project folder.



```
"use client";
export default function ErrorBoundary({
    error,
    reset,
}:{
    error:Error;
    reset:() =>void;
}) {
    return (
    <div>
        {error.message}<button onClick={reset}>Reset</button>
        </div>
    )
}
```

G:\Courses\Type Script\NextJs\1.More about Routing\routing\app\project\error.tsx • Untracked

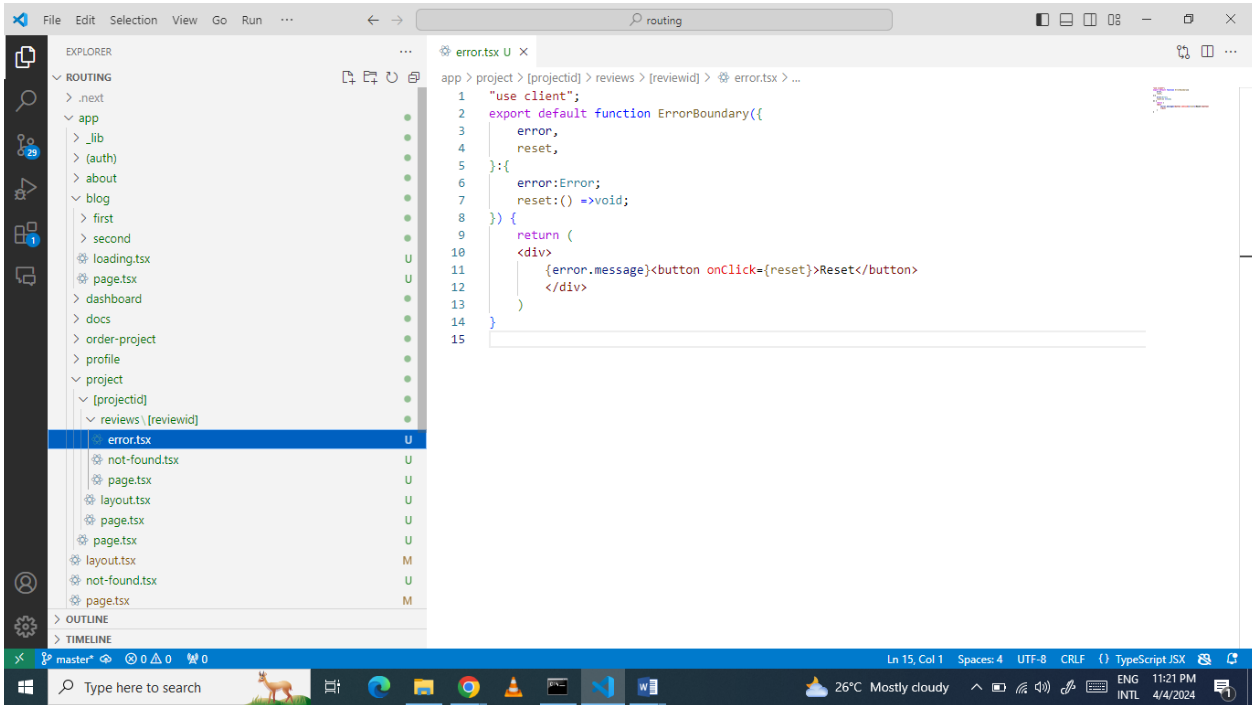
Go to the browser and navigate to http://localhost:3000/project/1/reviews/1 and refreshing the browser until an error occurs will display the familiar "Error loading review" message.


Next.js
localhost:3000/project/1/reviews/1
Header
error.message
Error loading review Reset
Footer
3 errors

As per the above screenshot of the browser, the entire project page will display the error message from the error.tsx file. The error originating from page.tsx in the review ID folder will now be managed by the error.tsx file in the project folder, which functions as the nearest error boundary.

If you put the error.tsx file back into the review ID folder and refresh the browser, the main project section in the product ID layout stays the same, and only the review part gets swapped with the UI in error.tsx. This shows a more detailed way of handling errors.

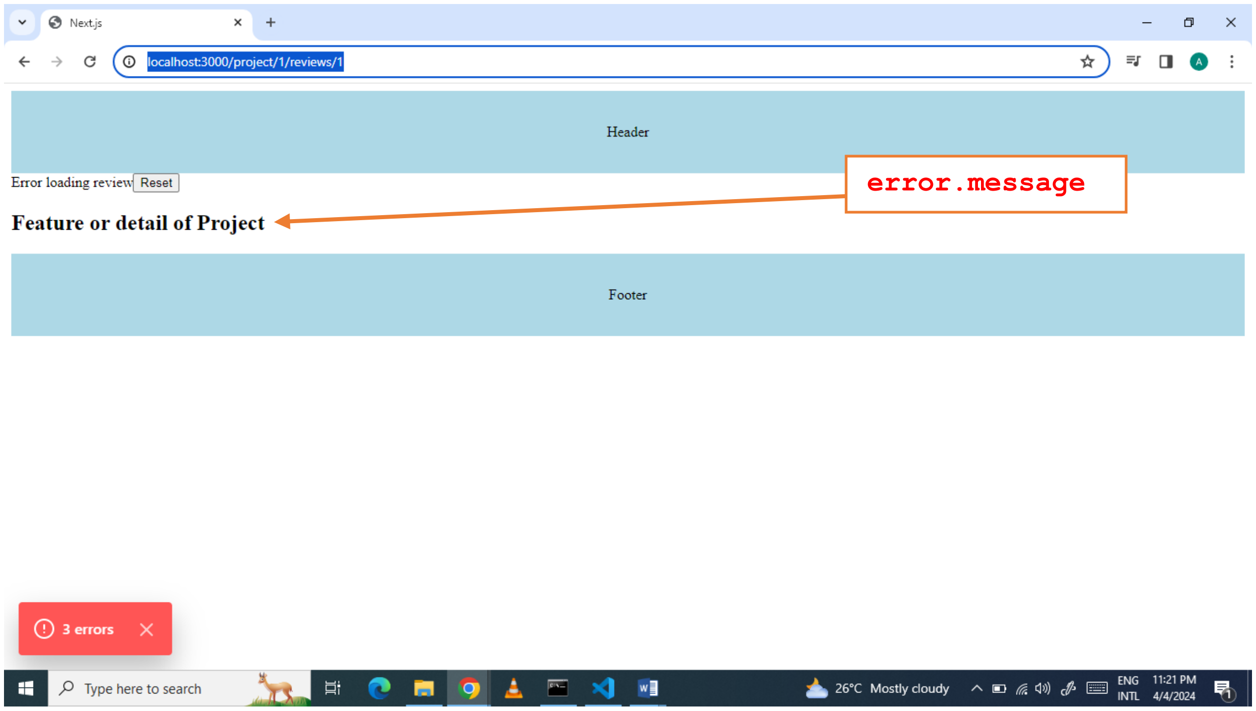
Move back the error.tsx file from the project folder to the review ID folder.



```tsx
"use client";
export default function ErrorBoundary({
    error,
    reset,
}:{
    error:Error;
    reset:() =>void;
}) {
    return (
    <div>
        {error.message}<button onClick={reset}>Reset</button>
        </div>
    )
}

```

Go to the browser and navigate to http://localhost:3000/project/1/reviews/1 and refreshing the browser until an error occurs will display the familiar "Error loading review" message.



This shows how errors work in nested routes. Where you place the error.tsx file is important because it decides how well you can control which parts of the UI are affected when there are errors deeper in the app. Putting error.tsx in the right spot helps you manage errors more carefully.