

Handling Errors in Layout in Next.js

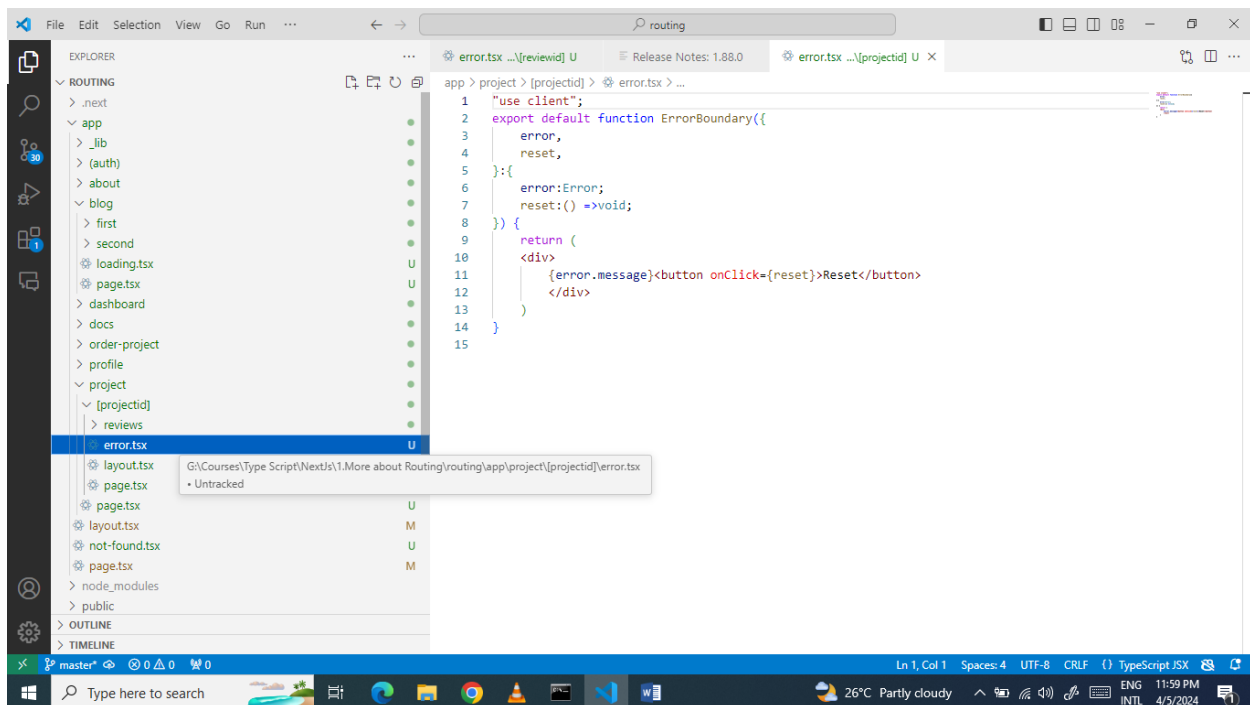
Step-by-Step Guide with Screenshots

Note:

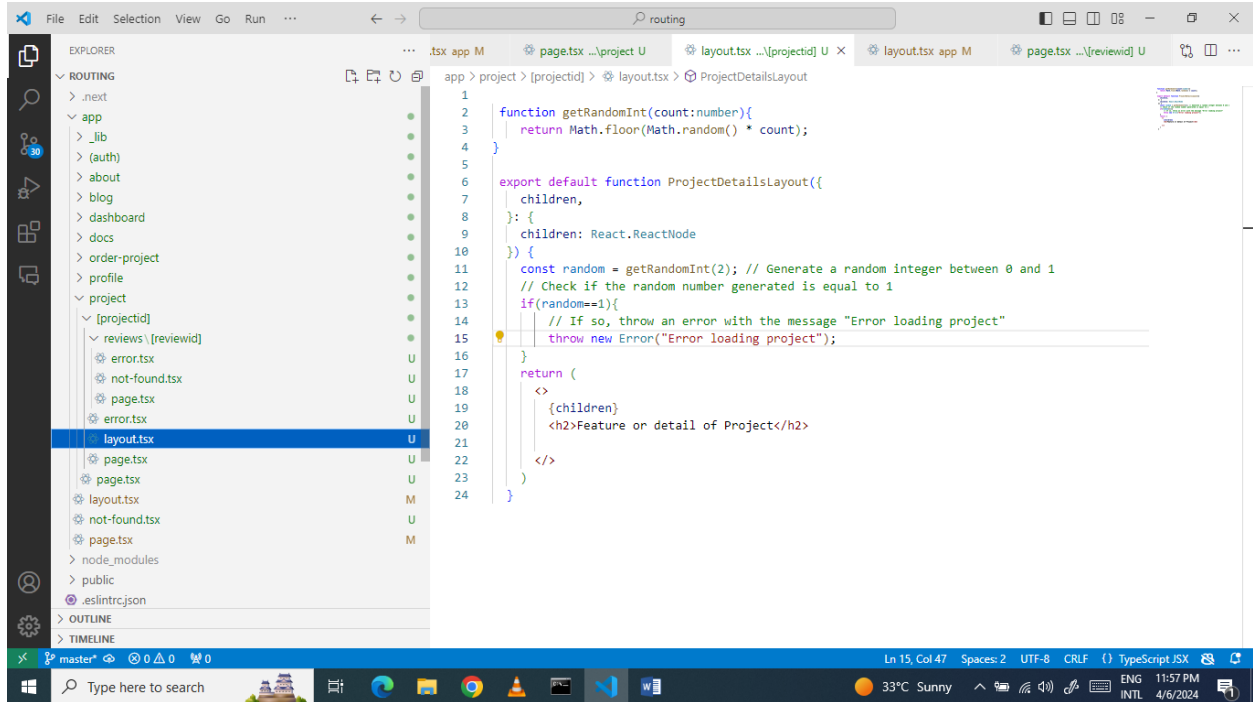
Continuing from the previous guide on Handling Errors in Next.js & Handling Error in Nested Routes in Next.js, this guide will discuss handling errors in layout in Next.js. In this guide, we will directly start practical implementation of handling errors in Layout in Next.js.

An `error.tsx` file can manage errors for all the smaller parts inside it (nested child segments). However, when it comes to a `layout.tsx` component in the same area, there's a difference. Errors occurring in this part aren't caught by the regular boundary because it's nested within the layout's part. So, if you examine how things are structured in that area, you'll see that the layout sits above the error boundary.

Copy the error.tsx file from the review ID folder to the projectid folder.

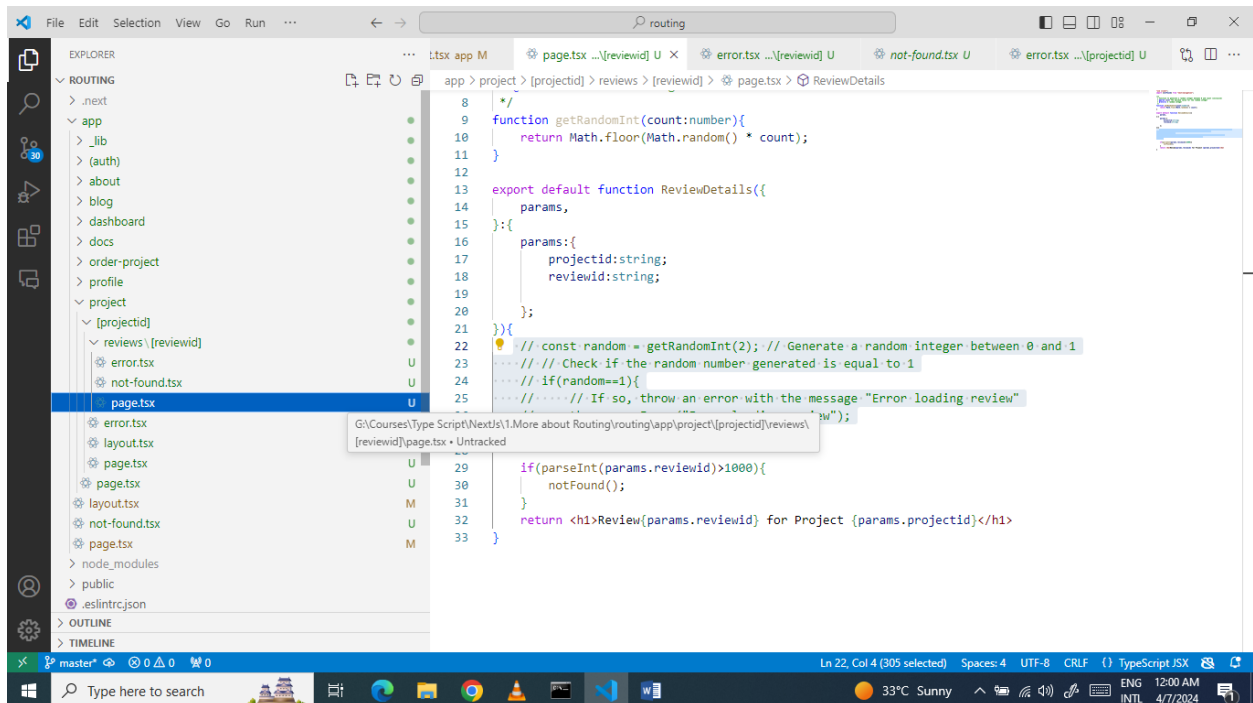


Update the layout.tsx file inside the projectid folder by adding the random number generator function, as highlighted in the screenshot below.



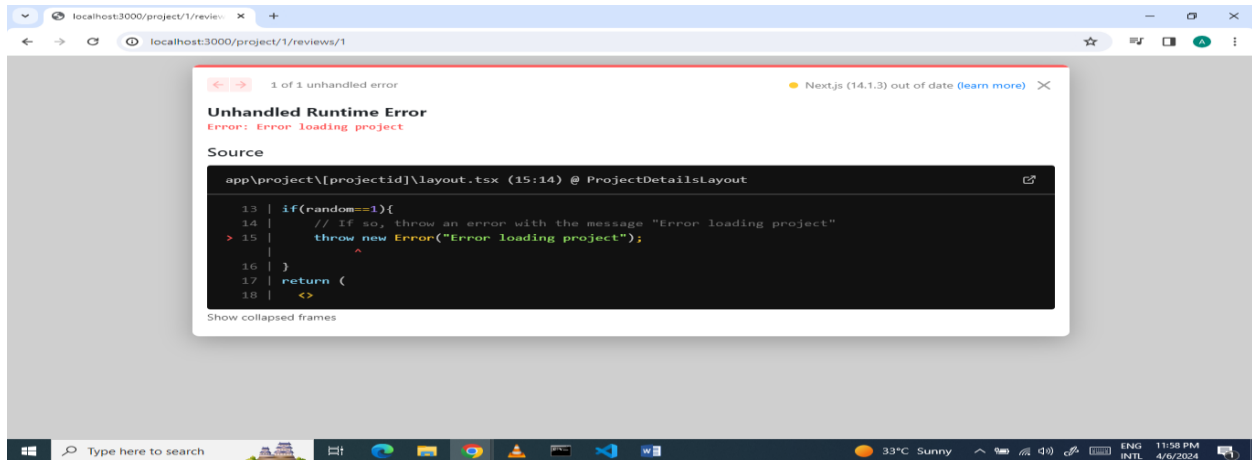
```
1
2 function getRandomInt(count:number){
3   return Math.floor(Math.random() * count);
4 }
5
6 export default function ProjectDetailsLayout({
7   children,
8 }): {
9   children: React.ReactNode
10 } {
11   const random = getRandomInt(2); // Generate a random integer between 0 and 1
12   // Check if the random number generated is equal to 1
13   if(random==1){
14     // If so, throw an error with the message "Error loading project"
15     throw new Error("Error loading project");
16   }
17   return (
18     <>
19     {children}
20     <h2>Feature or detail of Project</h2>
21   </>
22 )
23 }
24 }
```

Comment out the code in the page.tsx file inside the reviewid folder.



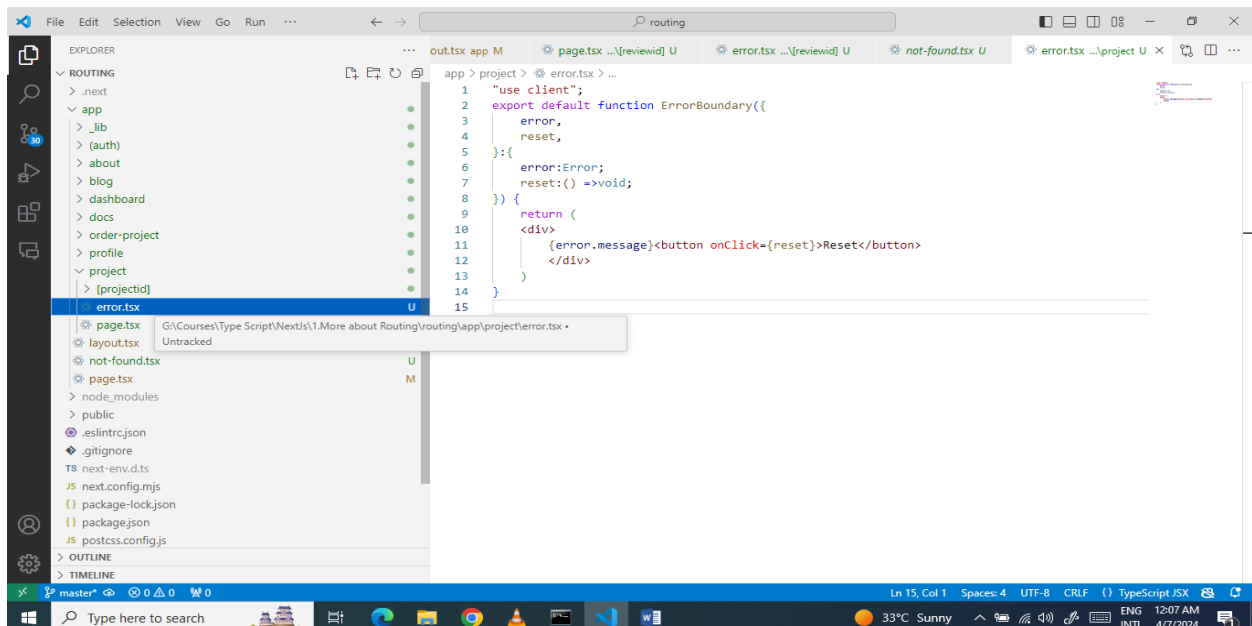
```
8 /*
9 function getRandomInt(count:number){
10   return Math.floor(Math.random() * count);
11 }
12
13 export default function ReviewDetails({
14   params,
15 }): {
16   params: {
17     projectid:string;
18     reviewid:string;
19   };
20 };
21 } {
22   // const random = getRandomInt(2); // Generate a random integer between 0 and 1
23   // Check if the random number generated is equal to 1
24   // if(random==1){
25     // If so, throw an error with the message "Error loading review"
26     // throw new Error("Error loading review");
27   }
28
29   if(parseInt(params.reviewid)>1000){
30     notFound();
31   }
32   return <h1>Review {params.reviewid} for Project {params.projectid}</h1>
33 }
```

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 project" message. We encounter the error message "Error loading project," but there's no error boundary to catch it. Consequently, our app is now experiencing a malfunction.



The error boundary won't manage errors thrown in a layout.tsx component within the same segment. To work around this, we should relocate the `error.tsx` file to the layout's parent segment. In this case, we move `error.tsx` from the projectid folder to the project folder.

Move the `error.tsx` file from the projectid folder to the project folder.



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 project" message.

