

Parallel Routes in Next.js

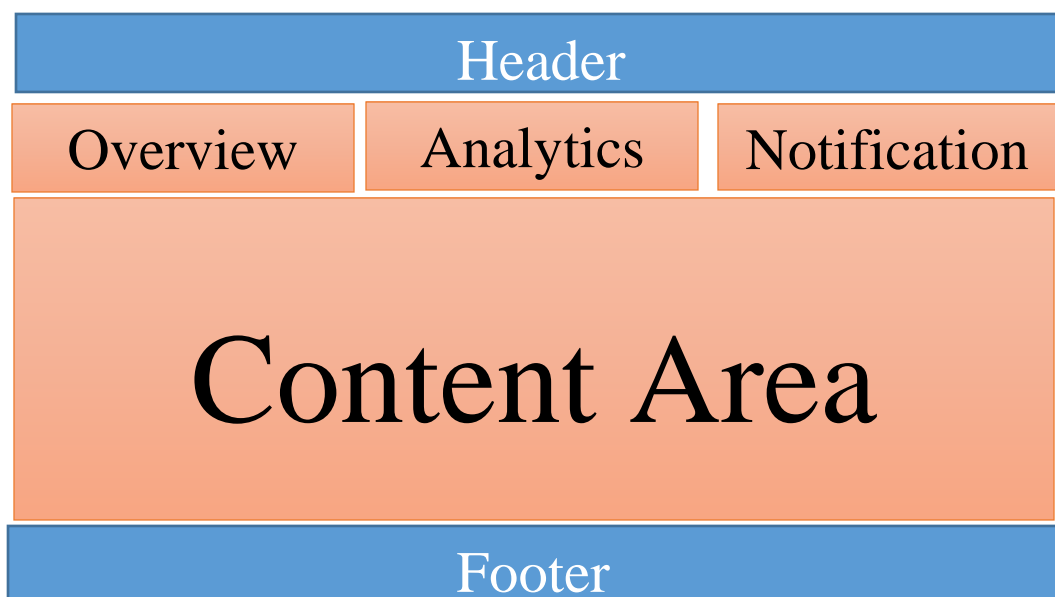
Parallel Routes in Next.js:

In Next.js, parallel routes refer to a feature that allows for the simultaneous rendering of multiple pages within the same layout. This means you can display different pages or components together, without the need for separate page navigations.

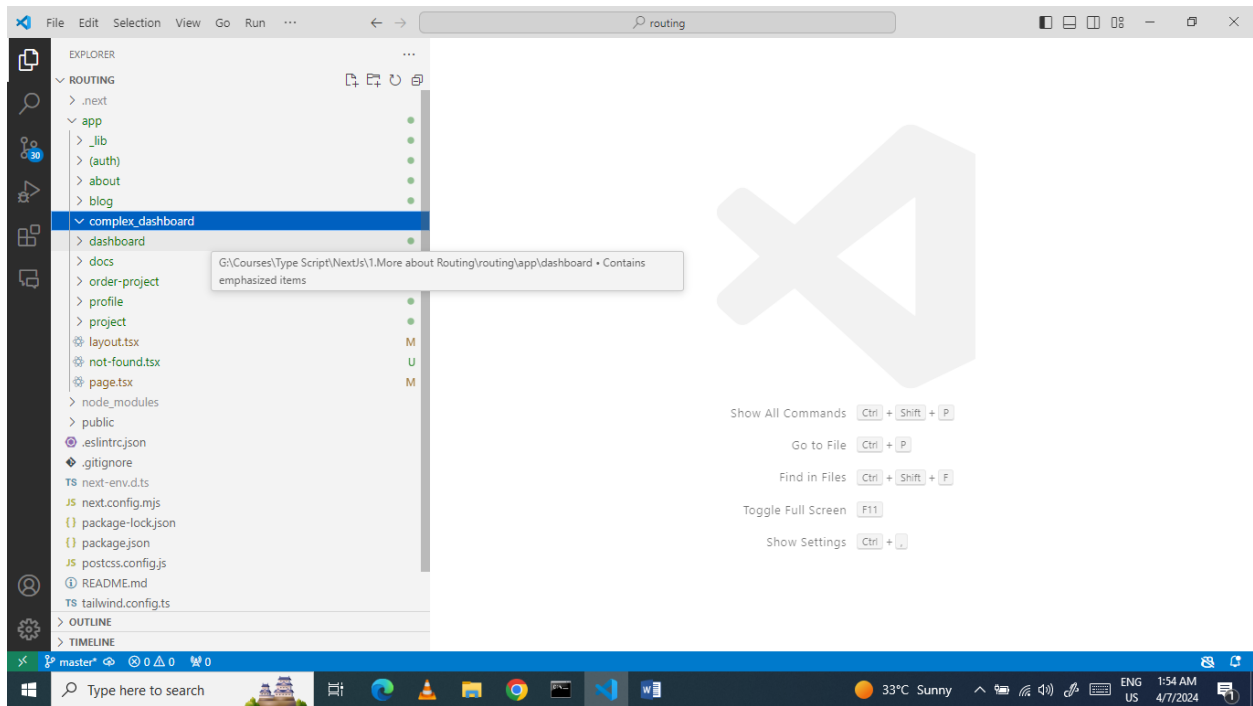
Parallel routes are implemented using a concept called slots. Slots help structure content in a modular fashion, allowing you to define distinct sections of your application. Each slot corresponds to a specific route or component, and they are organized within the same layout.

This feature is particularly useful when building complex user interfaces where multiple pieces of information need to be displayed together. It improves code organization and simplifies development by allowing different sections of the application to be developed independently.

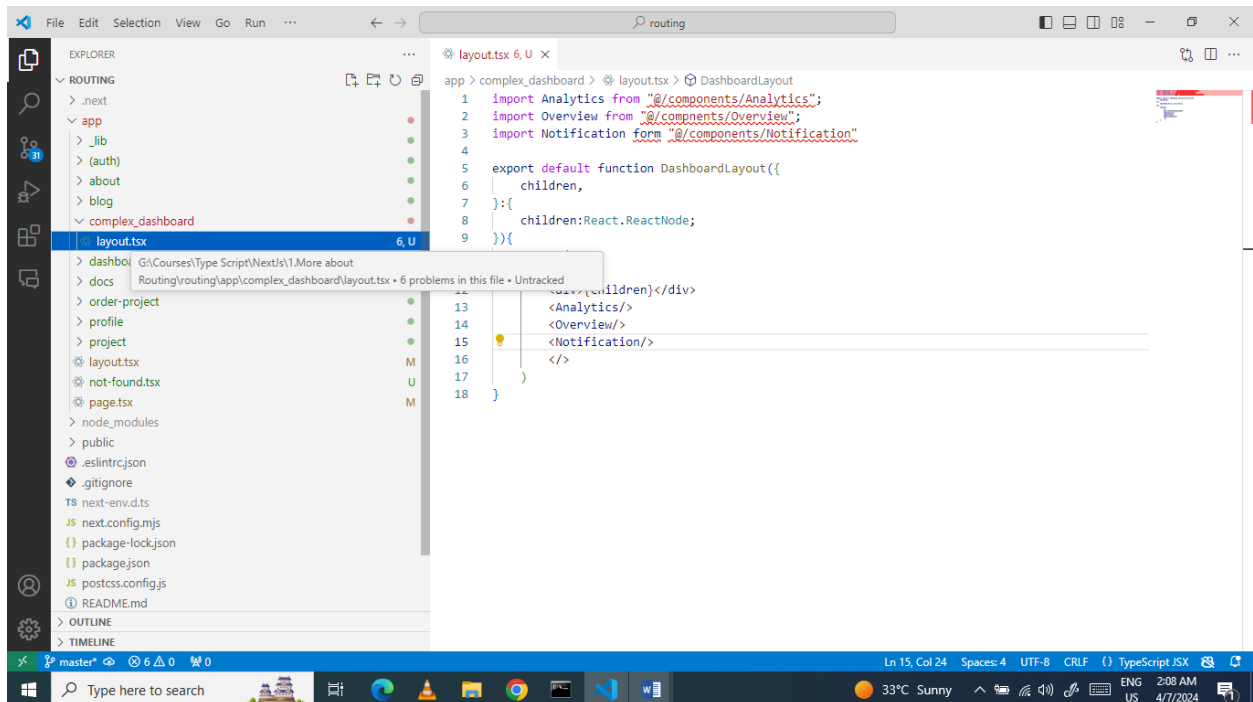
Overall, parallel routes in Next.js offer a way to create dynamic and interactive user interfaces by efficiently managing the rendering of multiple pages or components within a single layout. Below is the dashboard layout:



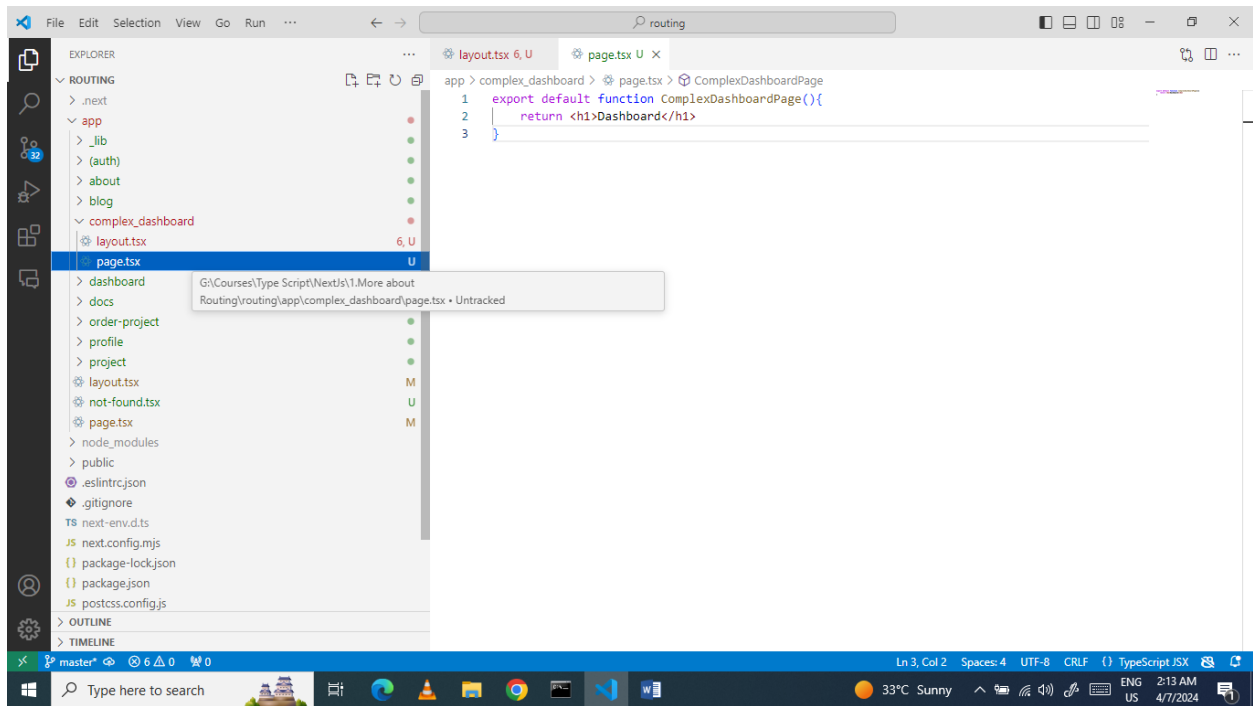
Create `complex_dashboard` folder inside the app folder.



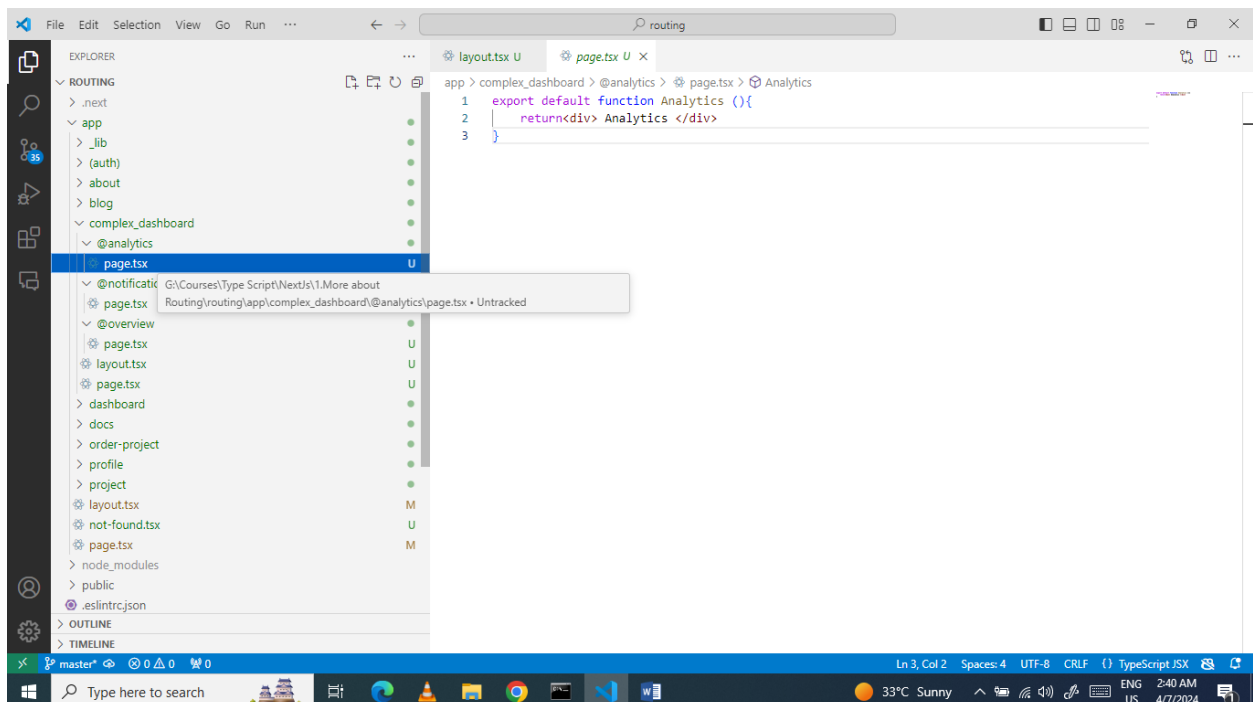
Inside the '`complex_dashboard`' folder, create a '`layout.tsx`' file using simple components. It might look something like this: three components for the three sections of the dashboard.

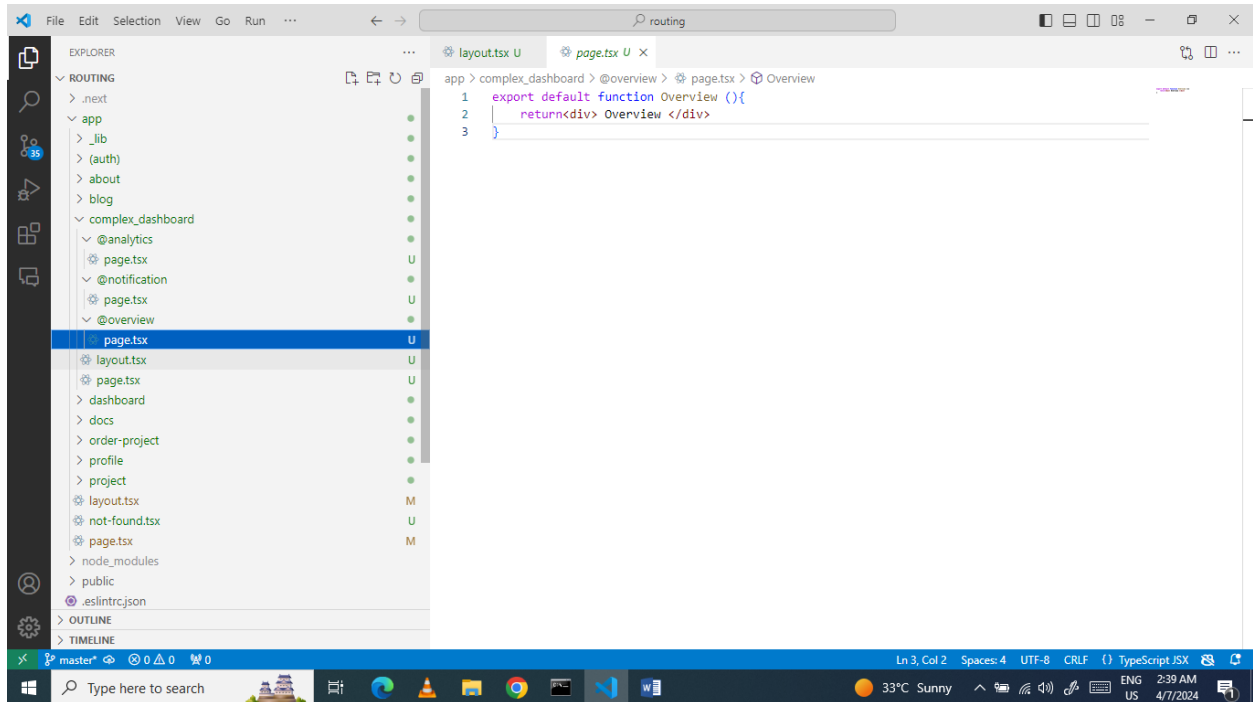
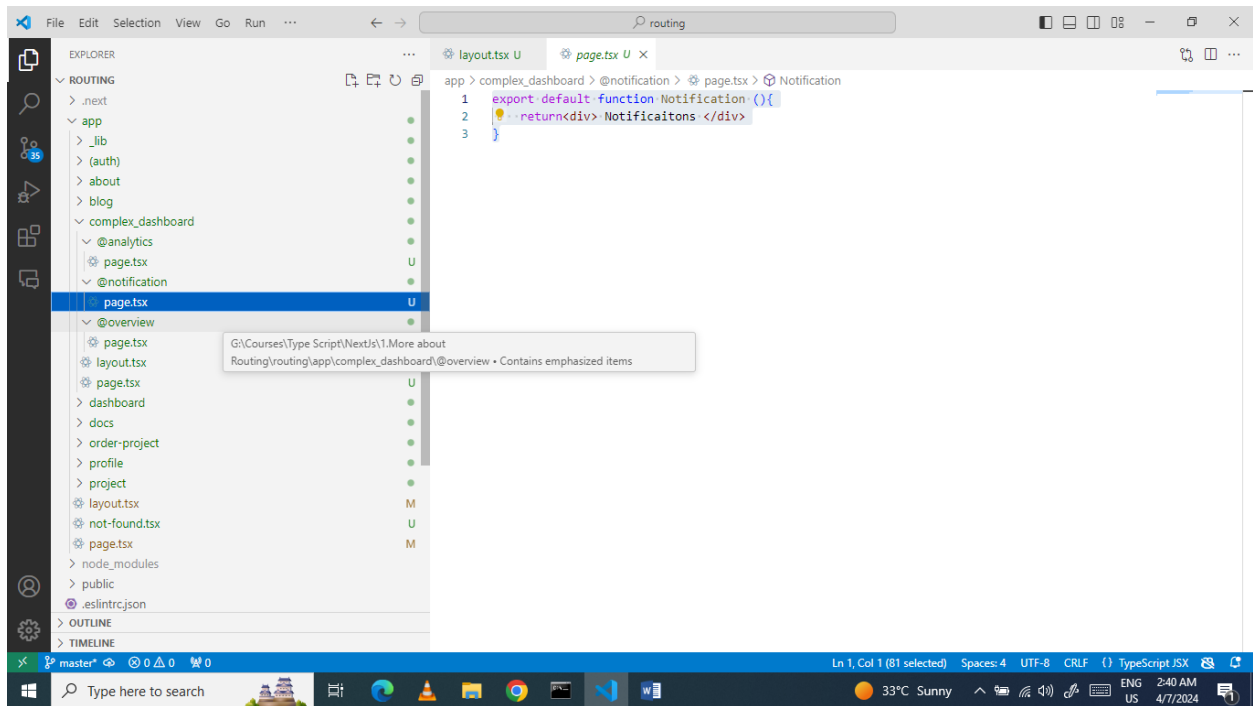


Create page.tsx file inside folder complex_dashboard

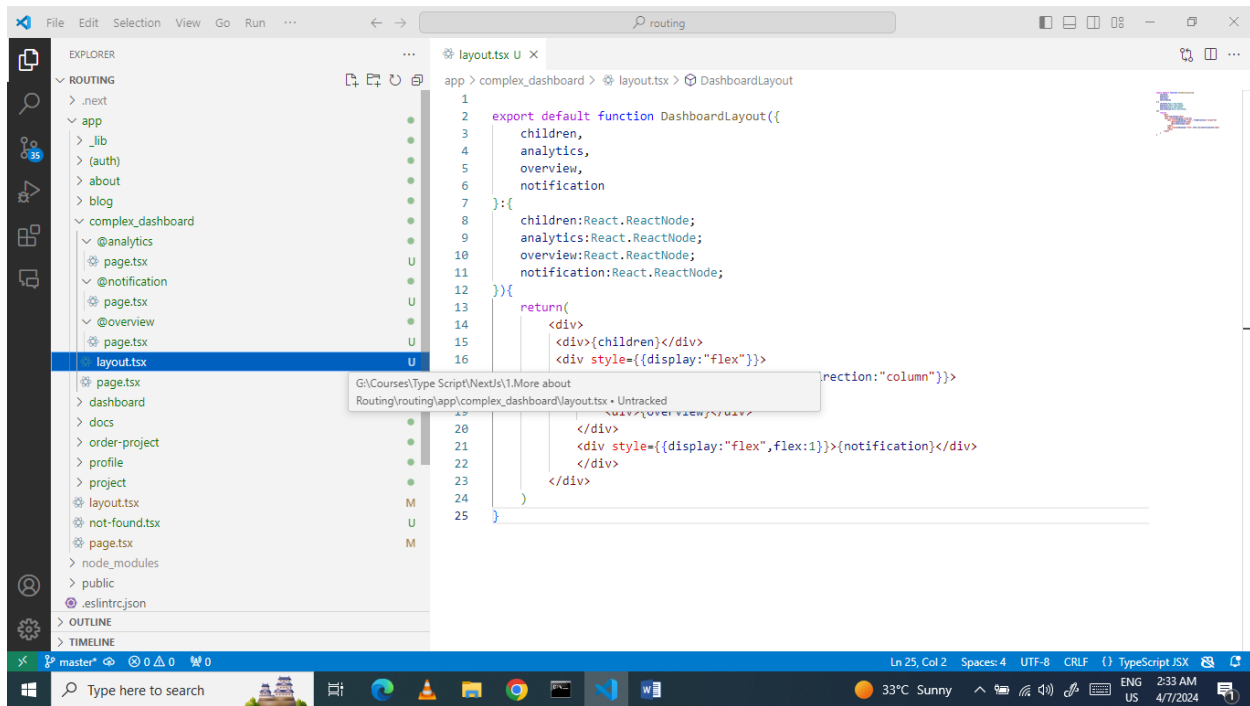


Inside the 'complex_dashboard' folder, create three folders named 'notification,' 'overview,' and 'analytics,' each starting with '@' (e.g., '@notification,' '@overview,' and '@analytics'). In each folder, create a 'page.tsx' file using simple components.





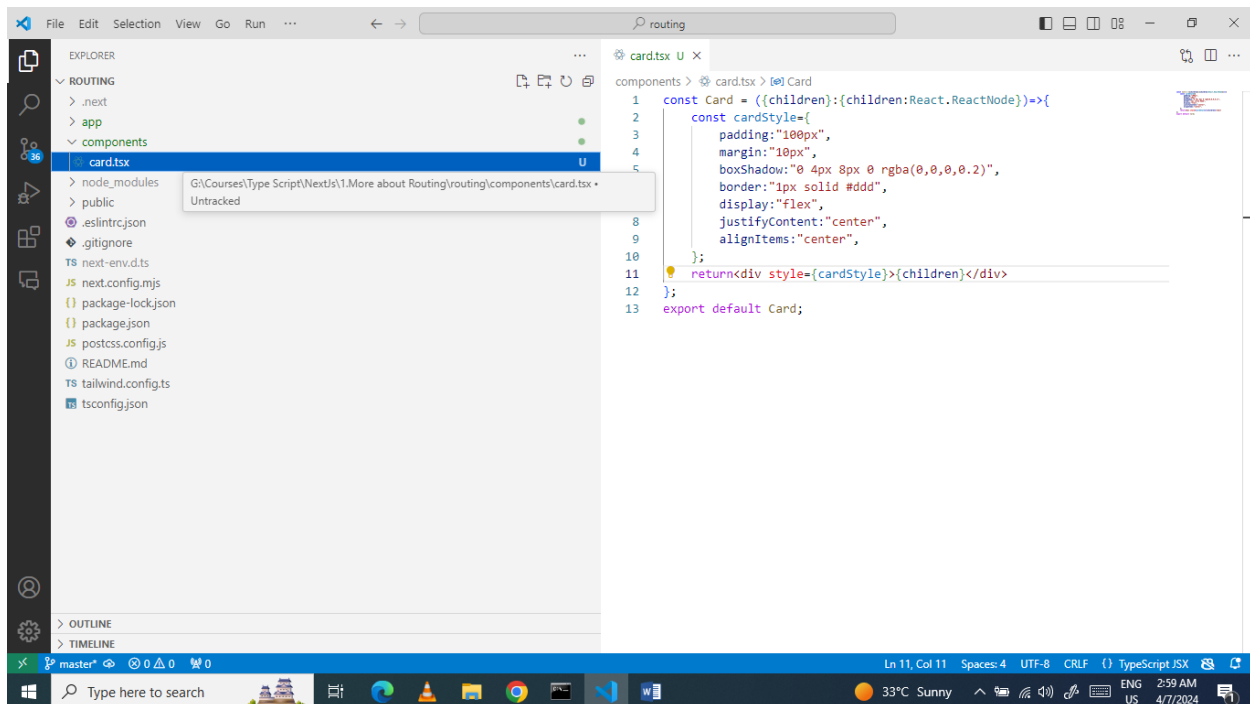
Update layout.tsx file inside complex_dashboard folder as per below snapshot



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The 'complex_dashboard' folder is expanded, showing subfolders like 'analytics', 'notification', 'overview', and 'layout.tsx'. The 'layout.tsx' file is selected and its content is displayed in the main editor. The code defines a 'DashboardLayout' function that takes children, analytics, overview, and notification as props. It returns a JSX element with a flex container for children and a notification component.

```
1 export default function DashboardLayout({
2   children,
3   analytics,
4   overview,
5   notification
6 }): {
7   children: React.ReactNode;
8   analytics: React.ReactNode;
9   overview: React.ReactNode;
10  notification: React.ReactNode;
11 } {
12   return(
13     <div>
14       <div>{children}</div>
15       <div style={{display:"flex"}}>
16         <div style={{display:"flex", flex:1}}>{notification}</div>
17       </div>
18     </div>
19   )
20 }
21
22
23
24
25
```

In the "src" (main folder, in my case "Routing") directory, create a new folder named "components." Inside the "components" folder, create a "card.tsx" file.

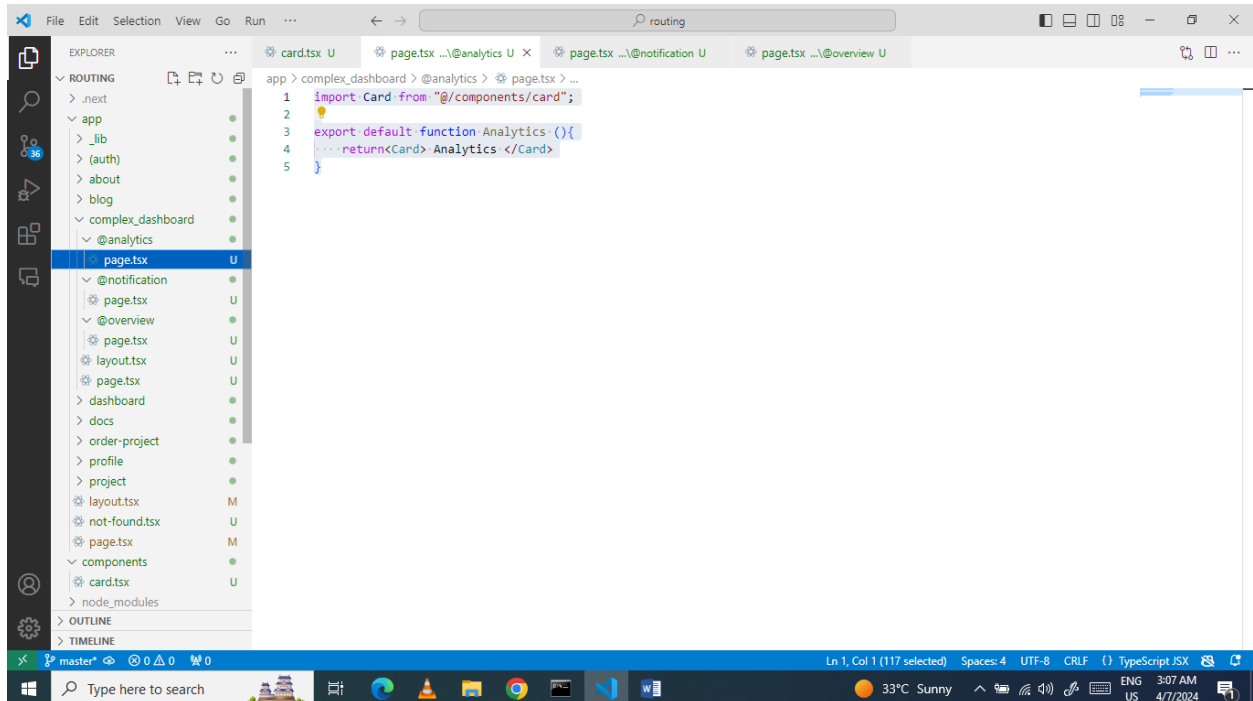


The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The 'components' folder is expanded, showing the 'card.tsx' file. The 'card.tsx' file is selected and its content is displayed in the main editor. The code defines a 'Card' component that takes children as a prop and returns a JSX element with a card style.

```
1 const Card = ({children}:{children:React.ReactNode})=>{
2   const cardStyle={
3     padding:"10px",
4     margin:"10px",
5     boxShadow:"0 4px 8px 0 rgba(0,0,0,0.2)",
6     border:"1px solid #ddd",
7     display:"flex",
8     justifyContent:"center",
9     alignItems:"center",
10  };
11   return<div style={cardStyle}>{children}</div>
12 };
13 export default Card;
```

Now, replace the `

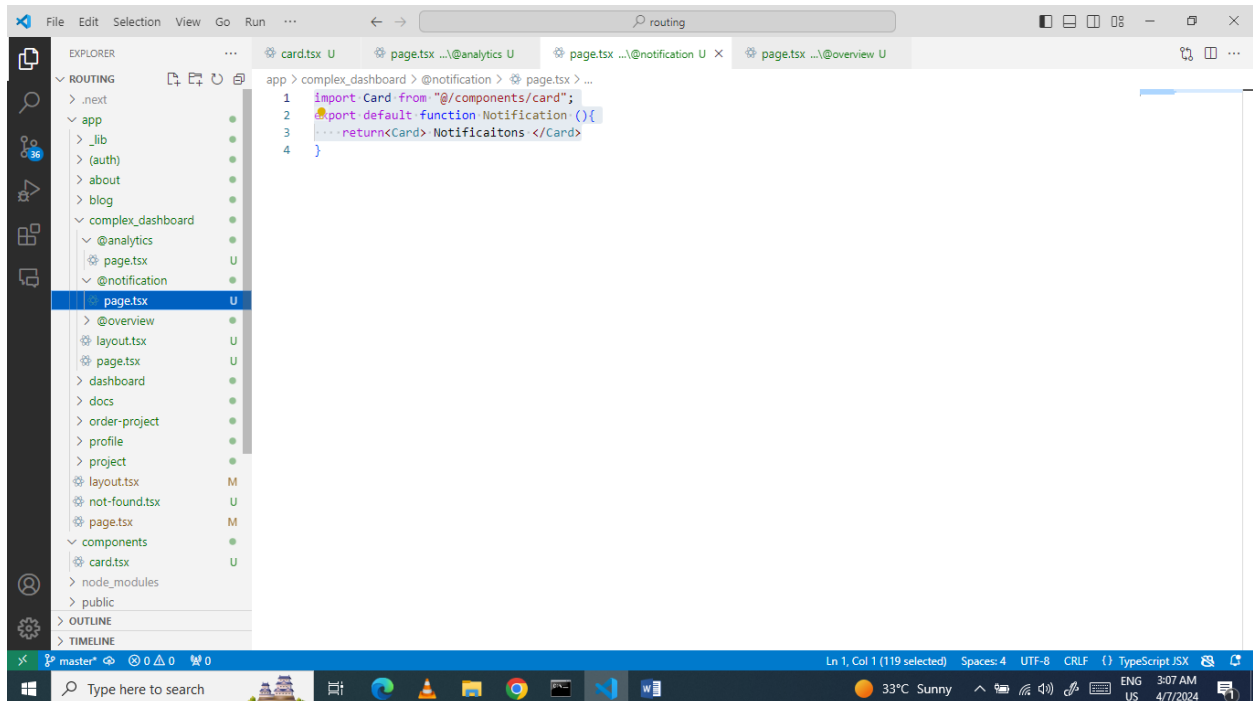
` tag in each file one by one with the Card component. See below screenshots for understanding.



This screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows a project structure with a 'complex_dashboard' folder containing '@analytics' and '@notification' subfolders. The '@analytics/page.tsx' file is selected. The main editor displays the following code:

```
1 import Card from "@components/card";
2
3 export default function Analytics() {
4   return <Card> Analytics </Card>;
5 }
```

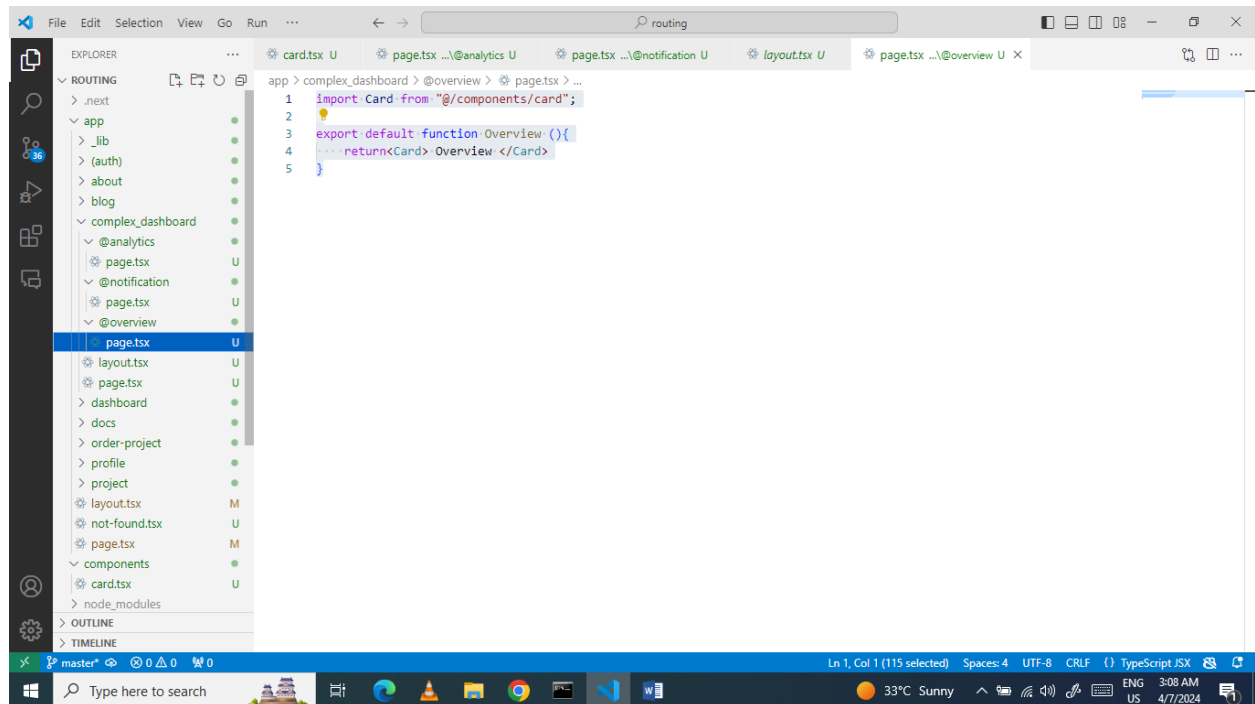
The status bar at the bottom indicates the file is 'Ln 1, Col 1 (117 selected)' and the encoding is 'UTF-8'.



This screenshot shows the Visual Studio Code editor with the file explorer on the left. The file explorer shows the same project structure as the previous screenshot. The '@notification/page.tsx' file is selected. The main editor displays the following code:

```
1 import Card from "@components/card";
2
3 export default function Notification() {
4   return <Card> Notification </Card>;
5 }
```

The status bar at the bottom indicates the file is 'Ln 1, Col 1 (119 selected)' and the encoding is 'UTF-8'.



"Go to the browser and navigate to http://localhost:3000/complex_dashboard to see the dashboard."

