

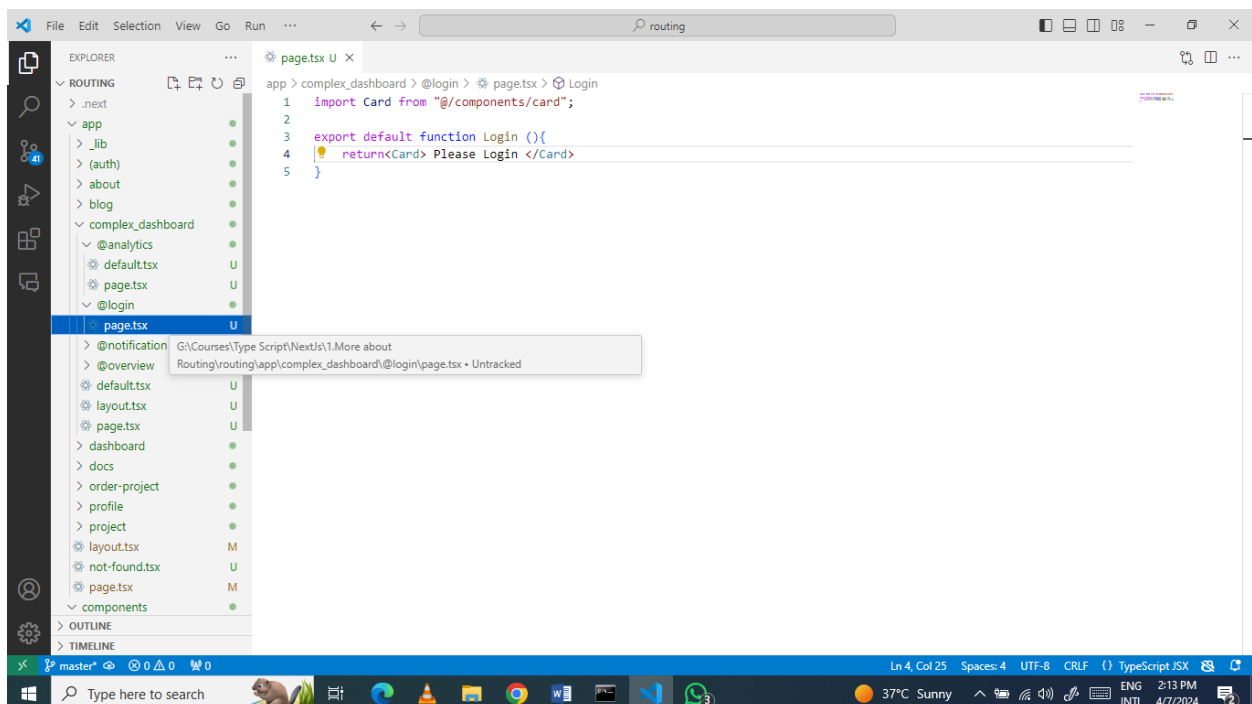
# Conditional Routing in Next.js

## Conditional Routing:

In Next.js, conditional routing refers to the ability to dynamically determine which page to render based on certain conditions or parameters. Next.js provides a flexible routing system that allows you to define routes for your application's pages. However, sometimes you may want to conditionally redirect users to different pages based on factors such as authentication status, user roles, or other application state.

Parallel routes offer a method for integrating conditional routing. For instance, based on the user's authentication status, you have the flexibility to display either the dashboard for authenticated users or a login page for those who are not authenticated. This functionality is significant as it enables the organization of code within the same URL, thereby improving adaptability and ease of maintenance.

Create a login slot inside `complex\_dashboard` with the name `@login`. Inside the `@login` folder, create a `page.tsx` file and update it as per the screenshots below.



```
File Edit Selection View Go Run ... routing
EXPLORER
  ROUTING
    .next
    app
      _lib
      (auth)
      about
      blog
      complex_dashboard
        @analytics
        default.tsx
        page.tsx
        @login
        page.tsx
      @notification
      @overview
        default.tsx
        layout.tsx
        page.tsx
      dashboard
      docs
      order-project
      profile
      project
      layout.tsx
      not-found.tsx
      page.tsx
    components
  OUTLINE
  TIMELINE
  page.tsx U
  G:\Courses\Type Script\Next.js\1.More about
  Routing\routing\app\complex_dashboard\@login\page.tsx • Untracked

app > complex_dashboard > @login > page.tsx > Login
1 import Card from "@/components/card";
2
3 export default function Login () {
4   return <Card> Please Login </Card>
5 }
```

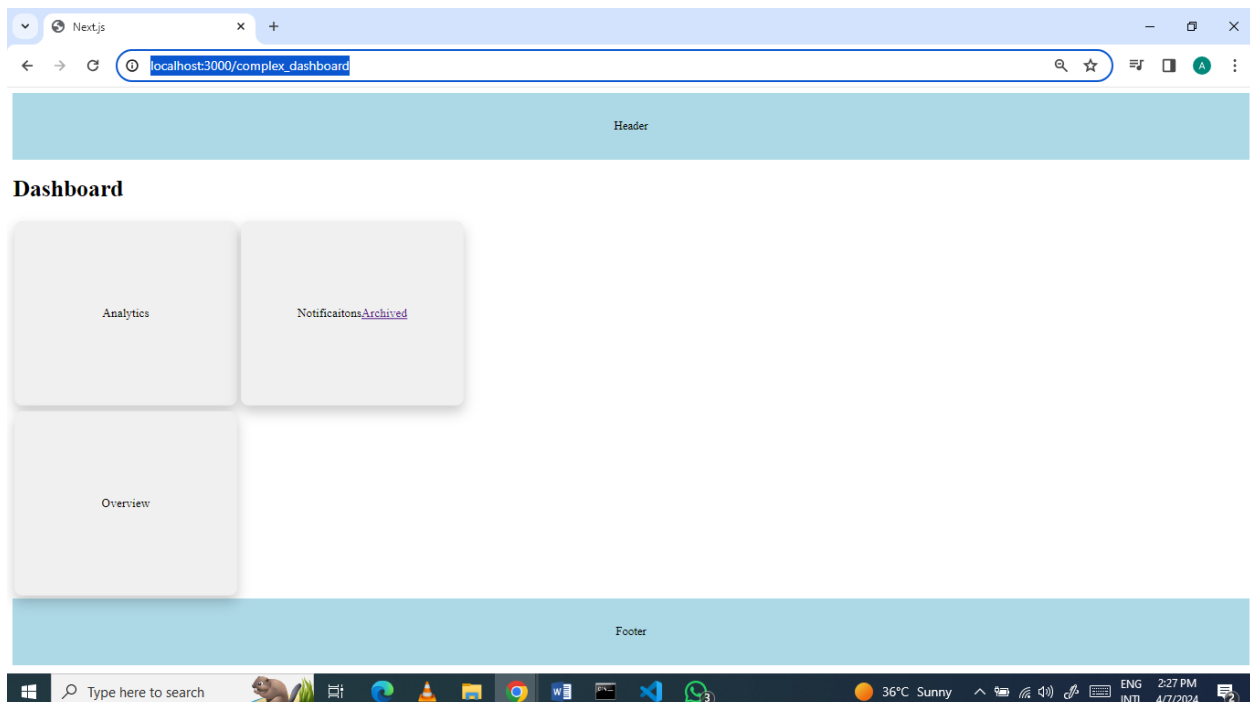
Ln 4, Col 25 Spaces: 4 UTF-8 CRLF TypeScript JSX

37°C Sunny 2:13 PM 4/7/2024

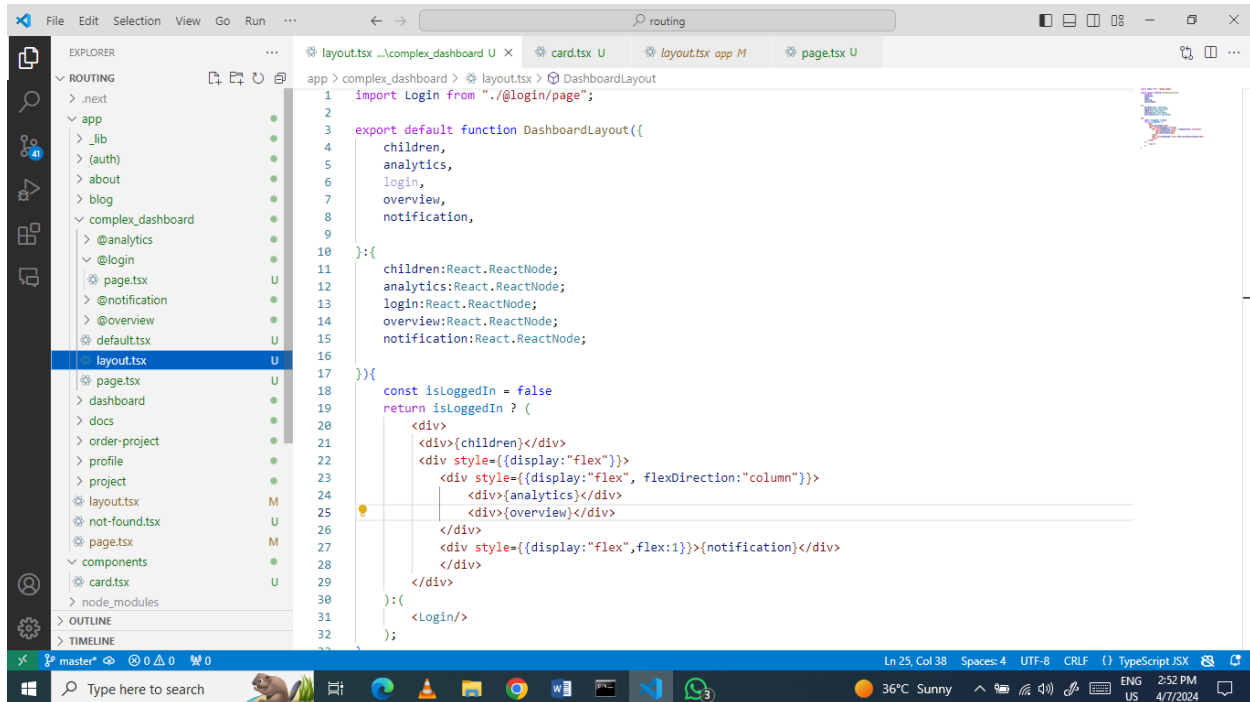
Open the layout.tsx file inside the complex\_dashboard folder. Update the layout.tsx file and set the isLoggedIn value to true as per the screenshots below.

```
1  export default function DashboardLayout({
2
3    children,
4    analytics,
5    login,
6    overview,
7    notification,
8
9  }): {
10
11    children: React.ReactNode;
12    analytics: React.ReactNode;
13    login: React.ReactNode;
14    overview: React.ReactNode;
15    notification: React.ReactNode;
16
17  }) {
18    const isLoggedIn = true
19    return isLoggedIn ? (
20      <div>
21        <div>{children}</div>
22        <div style={{display:"flex"}}>
23          <div style={{display:"flex", flexDirection:"column"}}>
24            <div>{analytics}</div>
25            <div>{overview}</div>
26          </div>
27          <div style={{display:"flex", flex:1}}>{notification}</div>
28        </div>
29      </div>
30    ) : (
31      <Login/>
32    );
33  }
```

"Go to the browser, navigate to [http://localhost:3000/complex\\_dashboard](http://localhost:3000/complex_dashboard), refresh the browser, and you will see the dashboard."

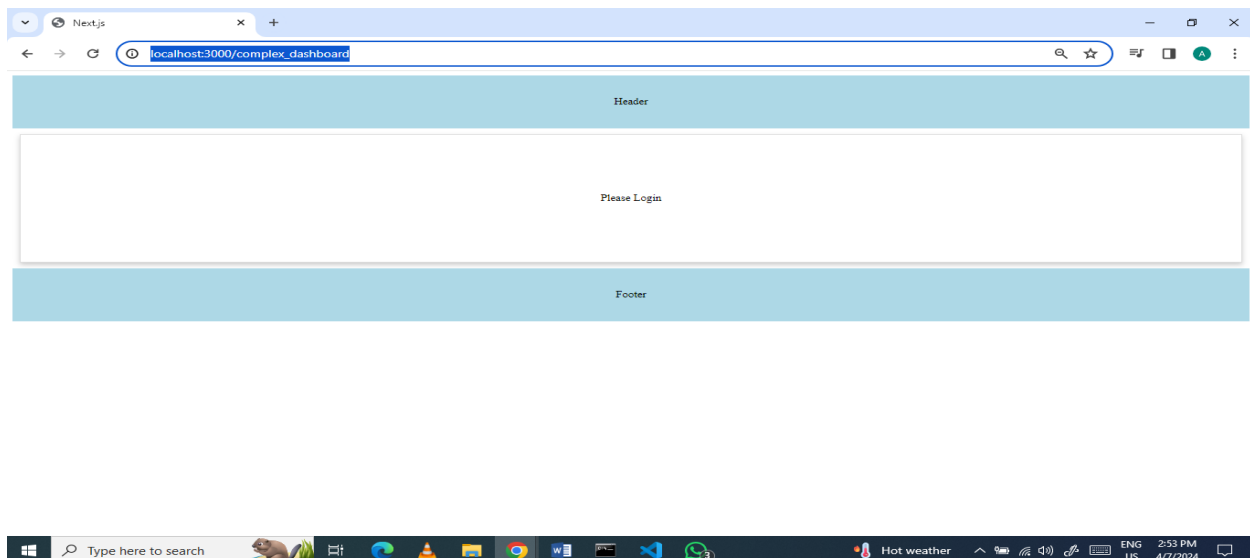


"Now open the `layout.tsx` file inside the `complex\_dashboard` folder and set the `isLoggedIn` value to false as shown in the screenshots below."



```
1 import Login from "../login/page";
2
3 export default function DashboardLayout({
4   children,
5   analytics,
6   login,
7   overview,
8   notification,
9 }) {
10   children: React.ReactNode;
11   analytics: React.ReactNode;
12   login: React.ReactNode;
13   overview: React.ReactNode;
14   notification: React.ReactNode;
15 }
16
17 ) {
18   const isLoggedIn = false
19   return isLoggedIn ? (
20     <div>
21       <div>{children}</div>
22       <div style={{display:"flex"}}>
23         <div style={{display:"flex", flexDirection:"column"}}>
24           <div>{analytics}</div>
25           <div>{overview}</div>
26         </div>
27         <div style={{display:"flex", flex:1}}>{notification}</div>
28       </div>
29     </div>
30   ) : (
31     <Login/>
32   );
33 }
```

"Now, go to the browser, navigate to [http://localhost:3000/complex\\_dashboard](http://localhost:3000/complex_dashboard), refresh the browser, and you will see the message 'please login'."



Parallel routes enable us to selectively display pages depending on specific conditions, effectively maintaining code separation within the same URL. Additionally, this login slot benefits from independent error and loading states, along with sub-navigation options to access "Sign up" or "Forgot password" routes.