Set up routing in the React app using react-router-dom to differentiate public and private routes. Implement protected routes for AdminDashboard and TestTakerDashboard that only authenticated users can access.

To solve the problem of implementing routing in a React app with react-router-dom to differentiate between public and private routes, follow these optimized step-by-step instructions:

**Step-by-Step Approach:**

**1. Install react-router-dom:**

* Open your terminal.
* Execute the command: npm install react-router-dom.

**2. Create Routing Setup:**

**In App.tsx:**

* Import BrowserRouter, Routes, Route, and any other necessary components from react-router-dom.
* Wrap the entire app with <BrowserRouter> to enable routing.
* Define public routes within the <Routes> component:
  + Route for the Home page (path /).
  + Route for the Login page (path /login).

**3. Implement Protected Routes:**

**Create PrivateRoute.tsx:**

* Define a PrivateRoute component.
* Inside the PrivateRoute component:
  + Check if the user is authenticated (this can be done by accessing the authentication status from your state management system, such as Redux, Context API, or any other).
  + If the user is authenticated, render the desired component (passed as a prop).
  + If the user is not authenticated, redirect to the Login page.

**In App.tsx:**

* Use the PrivateRoute component to define protected routes:
  + Route for AdminDashboard (path /admin-dashboard).
  + Route for TestTakerDashboard (path /test-taker-dashboard).

**4. Integrate Components:**

**Public Routes:**

* Ensure Home and Login components are created and imported.
* Integrate these components into the respective public routes.

**Private Routes:**

* Ensure AdminDashboard and TestTakerDashboard components are created and imported.
* Integrate these components into the respective protected routes using PrivateRoute.

**5. Styling:**

* Ensure all routes render clean and accessible pages by adding appropriate CSS.
* Use CSS modules, Styled Components, or any other styling approach preferred in your project.

**Deliverables:**

* Ensure your React app has:
  + Routing implemented in App.tsx.
  + A functional PrivateRoute component in PrivateRoute.tsx.
  + AdminDashboard and TestTakerDashboard components.
* Test the app to demonstrate the following behaviors:
  + Public routes (Home and Login pages) are accessible without authentication.
  + Unauthenticated users are redirected to the Login page when trying to access protected routes (AdminDashboard, TestTakerDashboard).
  + Authenticated users can access the AdminDashboard and TestTakerDashboard.

**Submission Guidelines:**

* Submit all modified files including App.tsx, PrivateRoute.tsx, and new dashboard components (AdminDashboard.tsx and TestTakerDashboard.tsx).
* Include evidence of testing such as screenshots or a recording showing both public and private route behavior.

By following these structured steps, you will be able to successfully implement routing and protect routes in your React application, ensuring that sensitive areas are only accessible to authenticated users.