Identify the routing approach: React routing.

The application's routing structure follows a well-organized and scalable approach, leveraging **React Router** for navigation and **lazy loading** for optimized performance. The routes are categorized into three main sections based on user roles:

* **Admin Routes** – Handles administrative functionalities like managing users, orders, payments, settings, and more.
* **Seller Routes** – Manages seller-specific operations such as product listings, orders, reviews, discounts, and financial transactions.
* **Waiter Routes** – Focused on waiter-related tasks, primarily managing orders and service operations.

Each category has its own **dedicated folder structure**, ensuring that route definitions remain modular and easy to maintain.

We can see   
/src

/routes

/admin

/appRoutes.js

/ addonRoutes.js

/ blogRoutes.js

...

/ usersRoutes.js

/ settingsRoutes.js

/seller

/ sellerAppRoutes.js

/ sellerOrderRoutes.js

/ sellerProductRoutes.js

...

/waiter

/ waiterAppRoutes.js

/ waiterOrderRoutes.js

/ index.js

The index.js in routing folder consolidates all routes This file exports AllRoutes which holds all the routing path and component objects of array from all the admin seller and waiter folder by using the operation … to extracting each array items and make a new array of all the routes. For centralization and managing the import or usage of the routing and accessing everything in one place which is AllRoutes.  
  
Each of these route files defines and exports an array of **route objects**, which include:

* **Path Definitions**: Specifies the route’s URL structure.
* **Lazy-Loaded Components**: Ensures that components are only loaded when needed.
* **Dynamic Routing**: Uses parameters (:id, :uuid, :type) for dynamic content loading.
* **Protected Routes**: Implements authentication and role-based access control   
  use or managed in a higher-order component also in a layout component.

Example in our code we used the role based access control in the app layout by checking the user’s role if it is admin seller or waiter.  
Also in this same component which is app-layout.js in the components folder its components will be rendered inside Outlet. That means Suspense from AppLayout is already covering your lazy-loaded route components. Which has the following uses:  
· Prevents Blank Screens – Without Suspense, if the component is still loading, users might see nothing. With Suspense, they see a loading indicator.

* · Better UX – Users get instant feedback instead of feeling like the app is stuck.
* · More Flexibility – You can define different loading indicators for different components.
* · Error Handling – It works well with Error Boundaries, so if a component fails to load, it won’t crash the whole app.

Instead of manually checking user.role in AppLayout, use a ProtectedRoute component that automatically restricts access. ProtectedRoute is not being used for role-based access. You have a basic ProtectedRoute, but it's only checking if a user exists. Instead, it should restrict access based on user roles.

Grouping Routes by User Type in AllRoutes

Right now, everything is merged into AllRoutes, making it hard to manage.

Instead, structure them like:

const AdminRoutes = [...];

const SellerRoutes = [...];

const WaiterRoutes = [...]; inside the index.js file in the routes folder.