**App.js**

import React, { useEffect } from "react";

import "./App.css";

import { BrowserRouter, Route, Routes } from "react-router-dom";

import Home from "./Pages/Home";

import Product from "./Pages/Product/Products";

import Login from "./Pages/Login/Login";

import SignUp from "./Pages/SignUp/SignUp";

import ForgetPassword from "./Pages/ForgetPassword";

import UpdatePassword from "./Pages/UpdatePassword";

import Checkout from "./Pages/CheckOut/Checkout";

import MyOrders from "./Pages/Orders/MyOrders";

import { useDispatch, useSelector } from "react-redux";

import { loadUserAction } from "./Redux/Actions/userAction";

import RestPassword from "./Pages/RestPassword";

import AdminHome from "./Pages/Admin/AdminHome";

import AddProduct from "./Pages/Admin/Product/AddProducts/AddProduct";

import AddCategory from "./Pages/Admin/Product/AddCategory/AddCategory";

import ViewProducts from "./Pages/Admin/Product/ViewsProducts/ViewProducts";

import OrderList from "./Pages/Admin/Orders/OrderList";

import UsersList from "./Pages/Admin/Users/UsersList";

import Family from "./Pages/Family";

import OrderDetails from "./Pages/Orders/OrderDetails";

import AllOrders from "./Pages/Orders/AllOrders";

import AddReviews from "./Pages/Reviews/AddReviews";

import AllReviews from "./Pages/Reviews/AllReviews";

import About from "./Pages/About/About";

import PageNotFound from "./Components/PageNotFound/PageNotFound";

import AdminProducts from "./Pages/Admin/Product/AdminProducts";

import UpdateProducts from "./Pages/Admin/Product/UpdateProducts/UpdateProducts";

import ViewCategory from "./Pages/Admin/Product/ViewCategory/ViewCategory";

import UpdateCategory from "./Pages/Admin/Product/UpdateCategory/UpdateCategory";

import AdminOrderDetails from "./Pages/Admin/Orders/OrderDetails";

import UpdateUser from "./Pages/Admin/Users/UpdateUser";

import ReviewsList from "./Pages/Admin/Reviews/ReviewsList";

const App = () => {

const { isAuthUser, user } = useSelector((state) => state.user);

const dispatch = useDispatch();

useEffect(() => {

dispatch(loadUserAction());

}, [dispatch]);

return (

<BrowserRouter>

<Routes>

<Route path="/" element={<Home />} />

<Route path="/Signup" element={!isAuthUser ? <SignUp /> : <Home />} />

<Route path="/Login" element={!isAuthUser ? <Login /> : <Home />} />

<Route

path="/RestPassword"

element={isAuthUser ? <UpdatePassword /> : <Login />}

/>

<Route

path="/Forget/Password"

element={!isAuthUser ? <ForgetPassword /> : <Home />}

/>

<Route path="/products" element={<Product />} />

<Route path="/products/:keyword" element={<Product />} />

<Route path="/products/category/:categoryId" element={<Product />} />

<Route path="/Order/Checkout" element={<Checkout />} />

<Route

path="/Order/MyOrder"

element={isAuthUser ? <AllOrders /> : <Login />}

/>

<Route

path="/Order/OrderDetails/:orderId"

element={isAuthUser ? <OrderDetails /> : <Login />}

/>

<Route

path="/reset-password/:id/:token"

element={!isAuthUser ? <RestPassword /> : <Login />}

/>

<Route

path="/review/add"

element={isAuthUser ? <AddReviews /> : <Login />}

/>

{/\* family Route \*/}

<Route

path="/user/:userId"

element={isAuthUser ? <Family /> : <Login />} />

<Route path="/reviews/all" element={<AllReviews />} />

<Route path="/about" element={<About />} />

{user && user.role === "Admin" ? (

<>

<Route path="/admin/home" element={<AdminHome />} />

{/\* Product Route \*/}

<Route path="/admin/products" element={<AdminProducts />} />

<Route path="/admin/add/products" element={<AddProduct />} />

<Route path="/admin/view/products" element={<ViewProducts />} />

<Route

path="/admin/update/products/:productId"

element={<UpdateProducts />}

/>

<Route path="/admin/add/category" element={<AddCategory />} />

<Route path="/admin/view/category" element={<ViewCategory />} />

<Route

path="/admin/update/category/:categoryId"

element={<UpdateCategory />}

/>

{/\* Orders Route \*/}

<Route path="/admin/view/orders" element={<OrderList />} />

<Route

path="/admin/update/order/:orderId"

element={<AdminOrderDetails />}

/>

{/\* Users Route \*/}

<Route path="/admin/view/users" element={<UsersList />} />

<Route path="/admin/update/:userId" element={<UpdateUser />} />

{/\* Reviews Route \*/}

<Route path="/admin/view/reviews" element={<ReviewsList />} />

</>

) : (

""

)}

<Route path="\*" element={<PageNotFound />} />

</Routes>

</BrowserRouter>

);

};

export default App;

**Store.js**

**import { createStore, applyMiddleware, combineReducers } from "redux";**

**import thunk from "redux-thunk";**

**import { composeWithDevTools } from "@redux-devtools/extension";**

**import {**

**loginReducer,**

**registerReducer,**

**logOutUser,**

**getUserReducer,**

**userPasswordUpdateReducer,**

**passwordResetEmailSendReducer,**

**userPasswordResetReducer,**

**getAllUserAdminReducer,**

**deleteUserAdminReducer,**

**updateUserRoleReducer,**

**} from "./Reducers/userReducers";**

**import {**

**getAllProductsReducer,**

**addProductsReducer,**

**deleteProductReducer,**

**getSingleProductReducer,**

**updateProductReducer,**

**} from "./Reducers/productReducer";**

**import { myCartProductReducer } from "./Reducers/cartReducer";**

**import {**

**addCategoryReducer,**

**deleteCategoryReducer,**

**getCategoryReducer,**

**updateCategoryReducer,**

**} from "./Reducers/categoryReducer";**

**import {**

**getAllOrdersAdminReducer,**

**getUpdateOrderAdminReducer,**

**getUserOrderDetailsReducer,**

**getUserOrderReducer,**

**} from "./Reducers/ordersReducer";**

**import {**

**deleteReviewsReducer,**

**getAllAdminReviewsReducer,**

**getAllReviewsReducer,**

**} from "./Reducers/reviewsReducer";**

**const rootReducer = combineReducers({**

**//User Reducer**

**register: registerReducer,**

**user: loginReducer,**

**userdetails: getUserReducer,**

**logOut: logOutUser,**

**passwordUpdate: userPasswordUpdateReducer,**

**sendRestPassMail: passwordResetEmailSendReducer,**

**passwordReset: userPasswordResetReducer,**

**getAllOrders: getUserOrderReducer,**

**getOrderDetails: getUserOrderDetailsReducer,**

**getAllReviews: getAllReviewsReducer,**

**userCart: myCartProductReducer,**

**// Product Reducers:**

**addProduct: addProductsReducer,**

**getAllProducts: getAllProductsReducer,**

**getSingleProduct: getSingleProductReducer,**

**deleteProduct: deleteProductReducer,**

**updateProduct: updateProductReducer,**

**//Category Reducer**

**addCategory: addCategoryReducer,**

**getAllCategory: getCategoryReducer,**

**deleteCategory: deleteCategoryReducer,**

**//Admin Reducer**

**adminAllOrders: getAllOrdersAdminReducer,**

**adminUpdateOrder: getUpdateOrderAdminReducer,**

**adminAllUsers: getAllUserAdminReducer,**

**adminDeleteUser: deleteUserAdminReducer,**

**adminUpdateUser: updateUserRoleReducer,**

**adminUpdateCategory: updateCategoryReducer,**

**adminGetReviews: getAllAdminReviewsReducer,**

**deleteReview: deleteReviewsReducer,**

**});**

**const initialState = {**

**userCart: {**

**cartItems: localStorage.getItem("userCart")**

**? JSON.parse(localStorage.getItem("userCart"))**

**: [],**

**},**

**};**

**const middleware = [thunk];**

**const store = createStore(**

**rootReducer,**

**initialState,**

**composeWithDevTools(applyMiddleware(...middleware))**

**);**

**export default**