AppNavbar.js :-

```
import React, { useState } from 'react';
import { Navbar, Nav, Container, NavDropdown, Modal, Button } from 'react-bootstrap';
import { LinkContainer } from 'react-router-bootstrap';
import { useAuth } from '../../context/AuthContext';

const AppNavbar = () => {
    const { user, logout } = useAuth();
    const [showLogoutModal, setShowLogoutModal] = useState(false);

    // 1. This is the new logic to check if the user is an admin
    const isAdmin = user && user.roles && user.roles.includes('ROLE_ADMIN');

    const handleLogout = () => {
        logout();
        setShowLogoutModal(false);
    };

    return (
        <>
            <Navbar bg="dark" variant="dark" expand="lg">
                <Container>
                    <LinkContainer to="/">
                        <Navbar.Brand>SnS Solutions</Navbar.Brand>
                    </LinkContainer>
                    <Navbar.Toggle aria-controls="basic-navbar-nav" />
                    <Navbar.Collapse id="basic-navbar-nav">
                        <Nav className="me-auto">
                            <LinkContainer to="/">
                                <Nav.Link>Home</Nav.Link>
                            </LinkContainer>
                            <LinkContainer to="/custom-pc">
                                <Nav.Link>Custom PC</Nav.Link>
                            </LinkContainer>
                            <LinkContainer to="/contact">
                                <Nav.Link>Contact</Nav.Link>
                            </LinkContainer>
                        </Nav>
                        <Nav>
                            {user ? (
                                <NavDropdown title={user.email} id="basic-nav-dropdown" align="end">
                                    {/* 2. We conditionally render the Admin link here */}
                                    {isAdmin && (
                                        <LinkContainer to="/admin-dashboard">
                                            <NavDropdown.Item>Admin Dashboard</NavDropdown.Item>
                                        </LinkContainer>
                                    )}
                                    <LinkContainer to="/my-orders">
                                        <NavDropdown.Item>My Orders</NavDropdown.Item>
                                    </LinkContainer>
                                    <NavDropdown.Divider />
                                    <NavDropdown.Item onClick={() => setShowLogoutModal(true)}>
                                        Logout
                                    </NavDropdown.Item>
                                </NavDropdown>
                            ) : (
                                <LinkContainer to="/login">
                                    <Nav.Link>Login</Nav.Link>
                                </LinkContainer>
                            )}
                        </Nav>
                    </Navbar.Collapse>
                </Container>
            </Navbar>

            {/* Your existing Logout Confirmation Modal remains unchanged */}
            <Modal show={showLogoutModal} onHide={() => setShowLogoutModal(false)} centered>
                <Modal.Header closeButton>
                    <Modal.Title>Confirm Logout</Modal.Title>
                </Modal.Header>
                <Modal.Body>Are you sure you want to log out?</Modal.Body>
                <Modal.Footer>
                    <Button variant="secondary" onClick={() => setShowLogoutModal(false)}>
                        Cancel
                    </Button>
                    <Button variant="danger" onClick={handleLogout}>
                        Logout
                    </Button>
                </Modal.Footer>
            </Modal>
        </>
    );
};
```

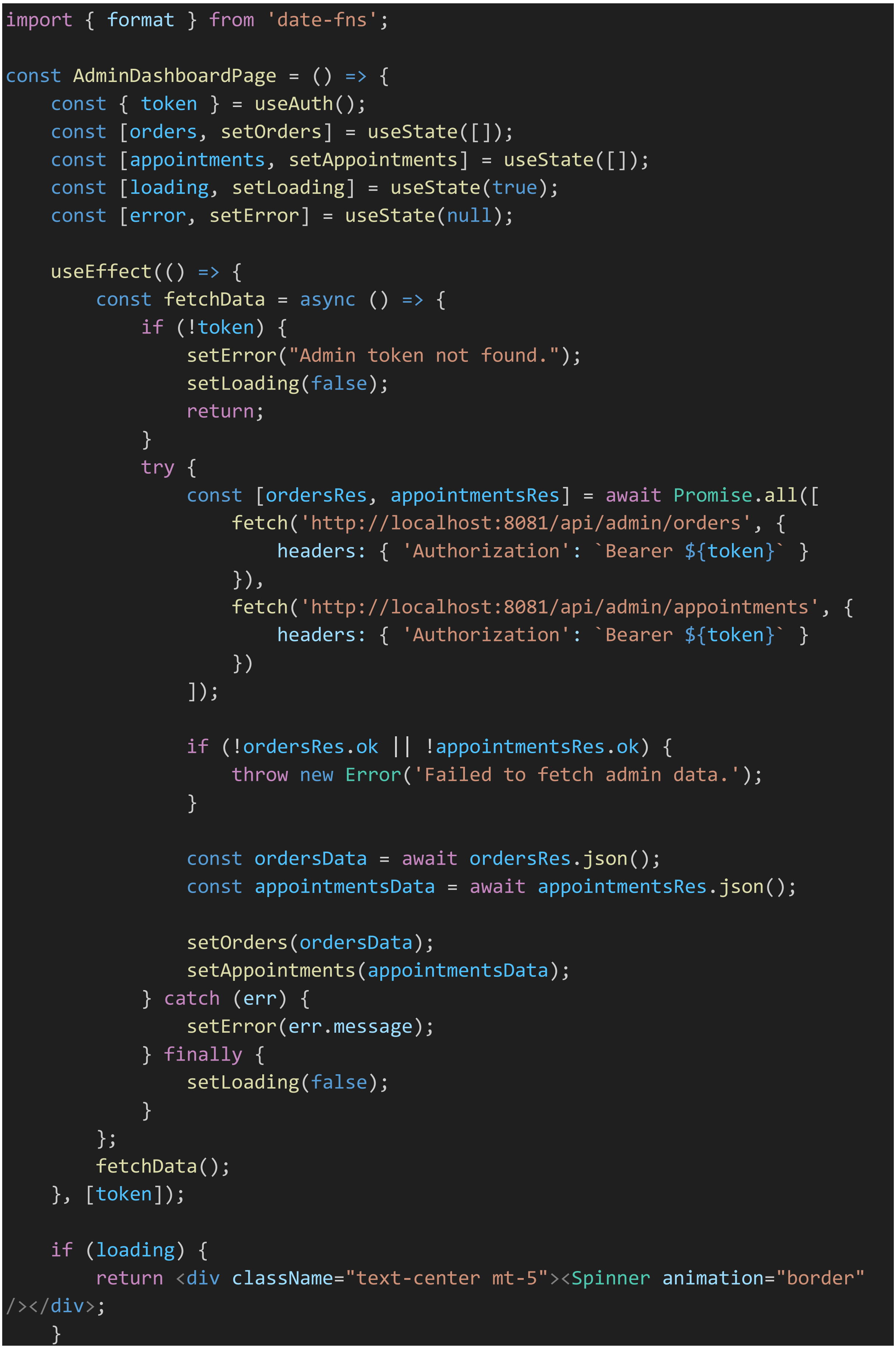
```

export default AppNavbar;

```

AuthContext.js :-

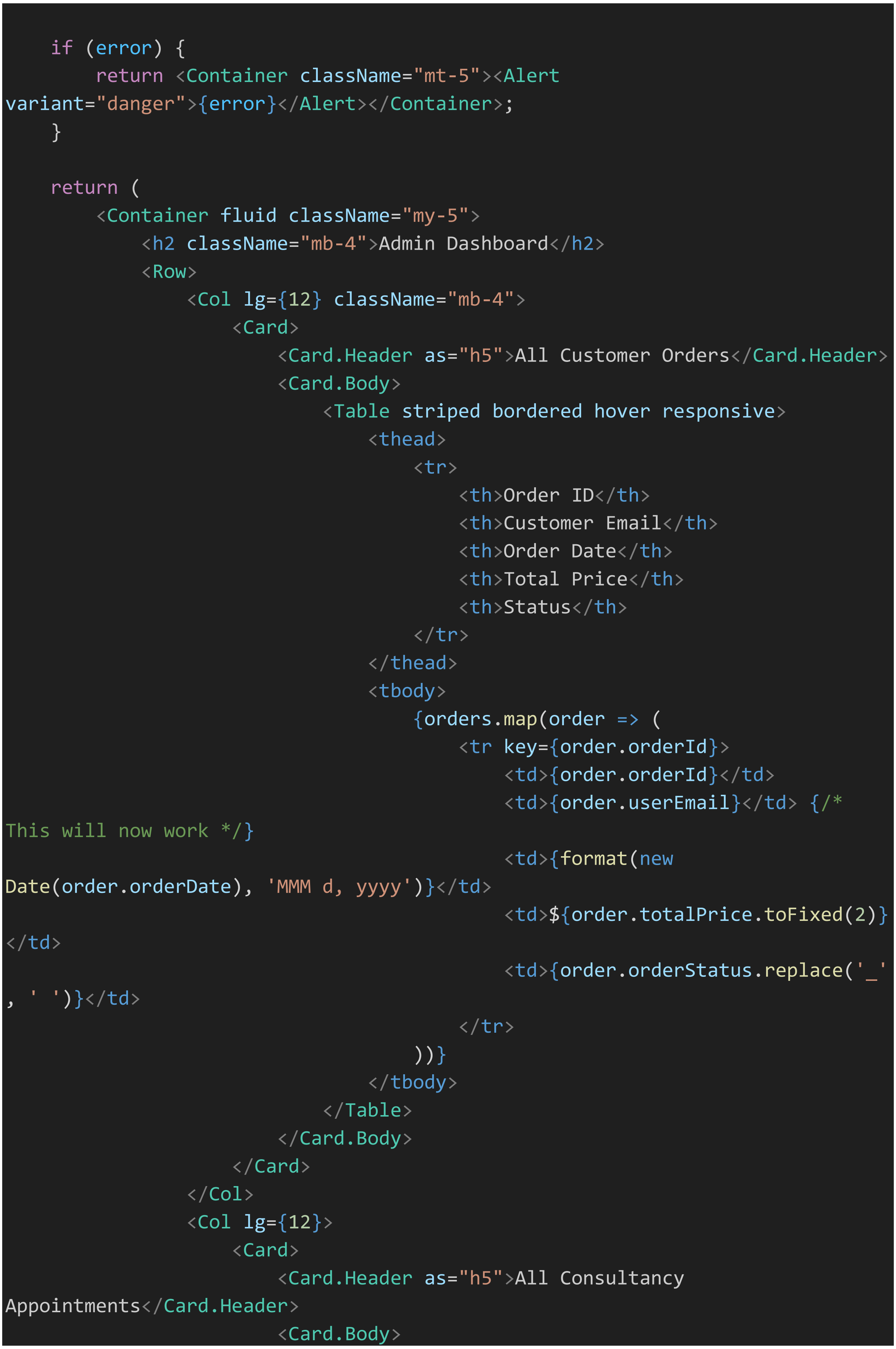
```
import React, { createContext, useState, useContext, useEffect, useCallback } from 'react';
import { useNavigate } from 'react-router-dom';
import { jwtDecode } from 'jwt-decode';

const AuthContext = createContext(null);

export const AuthProvider = ({ children }) => {
    // Initialize token from localStorage to keep user logged in
    const [token, setToken] = useState(() => localStorage.getItem('token'));
    const [user, setUser] = useState(null);
    const [error, setError] = useState('');
    const navigate = useNavigate();

    // Define logout with useCallback to prevent unnecessary re-renders
    const logout = useCallback(() => {
        localStorage.removeItem('token');
        setToken(null);
        setUser(null);
        navigate('/login');
    }, [navigate]); // navigate is a dependency since it's used in the function

    // This useEffect is the key. It runs when the app loads or when the token changes.
    useEffect(() => {
        if (token) {
            try {
                const decodedToken = jwtDecode(token);
                // Check if the token is expired
                if (decodedToken.exp * 1000 < Date.now()) {
                    logout(); // If expired, log out the user
                } else {
                    // THE FIX: Set the user state with both email AND roles from the token
                    setUser({ email: decodedToken.sub, roles: decodedToken.roles || [] });
                }
            } catch (e) {
                console.error("Invalid token:", e);
                logout(); // If the token is malformed, log out
            }
        } else {
            // If there's no token, ensure the user is logged out
            setUser(null);
        }
    }, [token, logout]); // Now includes logout in dependencies

    const login = async (email, password) => {
        setError('');
        try {
            const response = await
fetch('http://localhost:8081/api/auth/login', {
                method: 'POST',
                headers: { 'Content-Type': 'application/json' },
                body: JSON.stringify({ email, password }),
            });

            if (!response.ok) {
                const errorData = await response.json();
                throw new Error(errorData.error || 'Invalid email or
password');
            }

            const data = await response.json();
            localStorage.setItem('token', data.token);
            setToken(data.token); // This is the trigger that makes the
useEffect run
            navigate('/');
        } catch (err) {
            setError(err.message);
        }
    };

    const value = { user, token, error, login, logout };

    return <AuthContext.Provider
value={value}>{children}</AuthContext.Provider>;
};

export const useAuth = () => {
    return useContext(AuthContext);
};
```

AdminDashboardPage.js :-

```
import React, { useState, useEffect } from 'react';

import { useAuth } from '../context/AuthContext';
import { Container, Row, Col, Card, Spinner, Alert, Table } from 'react-
bootstrap';
```



```
import { format } from 'date-fns';

const AdminDashboardPage = () => {
    const { token } = useAuth();
    const [orders, setOrders] = useState([]);
    const [appointments, setAppointments] = useState([]);
    const [loading, setLoading] = useState(true);
    const [error, setError] = useState(null);

    useEffect(() => {
        const fetchData = async () => {
            if (!token) {
                setError("Admin token not found.");
                setLoading(false);
                return;
            }
            try {
                const [ordersRes, appointmentsRes] = await Promise.all([
                    fetch('http://localhost:8081/api/admin/orders', {
                        headers: { 'Authorization': `Bearer ${token}` }
                    }),
                    fetch('http://localhost:8081/api/admin/appointments', {
                        headers: { 'Authorization': `Bearer ${token}` }
                    })
                ]);

                if (!ordersRes.ok || !appointmentsRes.ok) {
                    throw new Error('Failed to fetch admin data.');
                }

                const ordersData = await ordersRes.json();
                const appointmentsData = await appointmentsRes.json();

                setOrders(ordersData);
                setAppointments(appointmentsData);
            } catch (err) {
                setError(err.message);
            } finally {
                setLoading(false);
            }
        };
        fetchData();
    }, [token]);

    if (loading) {
        return <div className="text-center mt-5"><Spinner animation="border"
/></div>;
    }
```



```
    if (error) {
        return <Container className="mt-5"><Alert
variant="danger">{error}</Alert></Container>;
    }

    return (
        <Container fluid className="my-5">
            <h2 className="mb-4">Admin Dashboard</h2>
            <Row>
                <Col lg={12} className="mb-4">
                    <Card>
                        <Card.Header as="h5">All Customer Orders</Card.Header>
                        <Card.Body>
                            <Table striped bordered hover responsive>
                                <thead>
                                    <tr>
                                        <th>Order ID</th>
                                        <th>Customer Email</th>
                                        <th>Order Date</th>
                                        <th>Total Price</th>
                                        <th>Status</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    {orders.map(order => (
                                        <tr key={order.orderId}>
                                            <td>{order.orderId}</td>
                                            <td>{order.userEmail}</td> {/*
This will now work */}
                                            <td>{format(new
Date(order.orderDate), 'MMM d, yyyy')}</td>
                                            <td>${order.totalPrice.toFixed(2)}
</td>
                                            <td>{order.orderStatus.replace('_'
, ' ')}</td>
                                        </tr>
                                    ))}
                                </tbody>
                            </Table>
                        </Card.Body>
                    </Card>
                </Col>
                <Col lg={12}>
                    <Card>
                        <Card.Header as="h5">All Consultancy
Appointments</Card.Header>
                        <Card.Body>
```

```
                            <Table striped bordered hover responsive>
                                <thead>
                                    <tr>
                                        <th>Appt ID</th>
                                        <th>Client Name</th>
                                        <th>Client Email</th>
                                        <th>Preferred Date</th>
                                        <th>Status</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    {appointments.map(appt => (
                                        <tr key={appt.appointmentId}>
                                            <td>{appt.appointmentId}</td>
                                            <td>{appt.clientName}</td>
                                            <td>{appt.clientEmail}</td>
                                            <td>{format(new
Date(appt.preferredDate), 'MMM d, yyyy')}</td>
                                            <td>{appt.status}</td>
                                        </tr>
                                    ))}
                                </tbody>
                            </Table>
                        </Card.Body>
                    </Card>
                </Col>
            </Row>
        </Container>
    );
};

export default AdminDashboardPage;
```

App.js :-

```
import React from 'react';

import { Routes, Route } from 'react-router-dom';
import { Container } from 'react-bootstrap';
import AppNavbar from './components/layout/AppNavbar'; // Corrected path for
consistency
import HomePage from './pages/HomePage';
import CustomPCPage from './pages/CustomPCPage';
import LoginPage from './pages/LoginPage';
import OrderHistoryPage from './pages/OrderHistoryPage';
import ContactPage from './pages/ContactPage';
import PrivacyPolicyPage from './pages/PrivacyPolicyPage'; // Renamed import
for clarity
```

```
import TermsPage from './pages/TermsPage';            // Renamed import for
clarity
import Footer from './components/Footer';
import RegisterPage from './pages/RegisterPage';
import AdminDashboardPage from './pages/AdminDashboardPage'; // 1. Import the
new Admin Page
import AdminRoute from './components/AdminRoute';       // 2. Import the Admin
Route guard

function App() {
  return (
    <div className="d-flex flex-column min-vh-100">
      <AppNavbar />
      <main className="flex-grow-1">
        <Container className="mt-4">
          <Routes>
            <Route path="/" element={<HomePage />} />
            <Route path="/custom-pc" element={<CustomPCPage />} />
            <Route path="/login" element={<LoginPage />} />
            <Route path="/register" element={<RegisterPage />} />
            <Route path="/my-orders" element={<OrderHistoryPage />} />
            <Route path="/contact" element={<ContactPage />} />
            <Route path="/privacy-policy" element={<PrivacyPolicyPage />} />
            <Route path="/terms-of-service" element={<TermsPage />} />

            {/* 3. Add the new, protected route for the Admin Dashboard */}
            <Route
              path="/admin-dashboard"
              element={
                <AdminRoute>
                  <AdminDashboardPage />
                </AdminRoute>
              }
            />
          </Routes>
        </Container>
      </main>
      <Footer />
    </div>
  );
}

export default App;
```