**ProtectedRoutev2.jsx:**

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

import { Navigate, useLocation } from 'react-router-dom';

import axiosInstancev2 from '../services/axiosInstancev2';

const ProtectedRoutev2 = ({ children }) => {

  const [isAuthenticated, setIsAuthenticated] = useState(null); // `null` to indicate loading

  const location = useLocation();

  useEffect(() => {

    const checkAuth = async () => {

      try {

        const response = await axiosInstancev2.get('/api/auth-check'); // Backend endpoint to validate token

        setIsAuthenticated(response.data.isAuthenticated);

      } catch (error) {

        setIsAuthenticated(false);

      }

    };

    checkAuth();

  }, []);

  if (isAuthenticated === null) {

    return <div>Loading...</div>; // Replace with a proper loader if needed

  }

  if (!isAuthenticated) {

    return <Navigate to="/adminlogin" replace />;

  }

  if (location.pathname === '/adminlogin' || location.pathname === '/') {

    return <Navigate to="/admindashboard" replace />;

  }

  return children;

};

export default ProtectedRoutev2;

**AdminDashboard.jsx:**

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

import './AdminDashboard.css';

import { FaUsers, FaChartBar, FaSignOutAlt, FaTags } from 'react-icons/fa'; // Importing icons

import { Pie, Bar } from 'react-chartjs-2';

import tupicon from '../../../assets/logo.png';

import irjpicon from '../../../assets/irjplogo.png';

import StudentListModal from '../../popups/StudentListModal';

import {

  Chart as ChartJS,

  Title,

  Tooltip,

  Legend,

  ArcElement,

  CategoryScale,

  LinearScale,

  BarElement,

} from 'chart.js';

import axiosInstance from '../../../services/axiosInstance';

ChartJS.register(CategoryScale, LinearScale, Title, Tooltip, Legend, ArcElement, BarElement);

const AdminDashboard = () => {

  const [activeSection, setActiveSection] = useState('Users'); // Track active section

  return (

    <div className="admin-dashboard-container">

      {/\* Sidebar \*/}

      <div className="adminsidebar">

        <nav>

          <ul>

            <li className="iconli">

              <div className="imgcontaineradmin">

                <img className="irjp" src={irjpicon} alt="" />

                <img className="tup" src={tupicon} alt="" />

              </div>

            </li>

            <li

              onClick={() => setActiveSection('Users')}

              className={activeSection === 'Users' ? 'active' : ''}

            >

              <FaUsers className="icon" />

              <span className="text">Users</span>

            </li>

            <li

              onClick={() => setActiveSection('Tags')}

              className={activeSection === 'Tags' ? 'active' : ''}

            >

              <FaTags className="icon" />

              <span className="text">Best</span>

            </li>

            <li>

              <FaSignOutAlt className="icon" />

              <span className="text">Log Out</span>

            </li>

          </ul>

        </nav>

      </div>

      {/\* Main Content \*/}

      <div className="admindashboard-content">

        {activeSection === 'Users' && <UsersSection />}

        {activeSection === 'Tags' && <TagChart />}

      </div>

    </div>

  );

};

// Users Section Component

const UsersSection = () => {

  const [selectedType, setSelectedType] = useState('Students'); // Track user type

  const [usersData, setUsersData] = useState([]); // Store fetched users

  const [loading, setLoading] = useState(false); // Loading state

  const [userStats, setUserStats] = useState({ studentCount: 0, employerCount: 0 }); // User stats for the pie chart

  const handleDropdownChange = (event) => {

    setSelectedType(event.target.value);

  };

  // Fetch users and user stats whenever the selected type changes

  useEffect(() => {

    const fetchUsers = async () => {

      setLoading(true);

      try {

        const response = await axiosInstance.get(`/api/users?type=${selectedType}`);

        if (response.data.success) {

          setUsersData(response.data.users);

        }

      } catch (error) {

        console.error('Error fetching users:', error);

      } finally {

        setLoading(false);

      }

    };

    const fetchUserStats = async () => {

      try {

        const response = await axiosInstance.get('/api/user-stats');

        if (response.data.success) {

          setUserStats({

            studentCount: response.data.studentCount,

            employerCount: response.data.employerCount,

          });

        }

      } catch (error) {

        console.error('Error fetching user stats:', error);

      }

    };

    fetchUsers();

    fetchUserStats();

  }, [selectedType]);

  const handleDelete = async (userId) => {

    const confirmDelete = window.confirm("Are you sure you want to delete this user?");

    if (!confirmDelete) return;

    try {

      const response = await axiosInstance.delete(`/api/manage-users/${userId}?type=${selectedType}`);

      if (response.data.success) {

        alert("User deleted successfully!");

        // Remove the user from the UI

        setUsersData((prevUsers) => prevUsers.filter((user) => user.\_id !== userId));

      } else {

        alert("Failed to delete the user.");

      }

    } catch (error) {

      console.error("Error deleting user:", error);

      alert("An error occurred while deleting the user.");

    }

  };

  // Pie chart data

  const chartData = {

    labels: ['Students', 'Employers'],

    datasets: [

      {

        data: [userStats.studentCount, userStats.employerCount],

        backgroundColor: [

          'rgba(54, 162, 235, 0.6)',  // Light blue with opacity

          'rgba(255, 99, 132, 0.6)',  // Light red with opacity

        ],

        hoverBackgroundColor: [

          'rgba(54, 162, 235, 0.9)',  // Darker blue on hover

          'rgba(255, 99, 132, 0.9)',  // Darker red on hover

        ],

        borderColor: ['#36A2EB', '#FF6384'],  // Strong border color

        borderWidth: 0, // Slightly thicker border for a sharper look

        hoverBorderWidth: 0, // Thicker border on hover

      },

    ],

  };

  // Render table dynamically based on selected type

  const renderTable = () => {

    if (loading) return <p>Loading...</p>;

    if (usersData.length === 0) return <p>No users found.</p>;

    if (selectedType === 'Students') {

      return (

        <table>

          <thead>

            <tr>

              <th>Name</th>

              <th>Email</th>

              <th>Contact</th>

              <th></th>

            </tr>

          </thead>

          <tbody>

            {usersData.map((user, index) => {

              const firstName = user?.profileDetails?.firstName || 'N/A';

              const lastName = user?.profileDetails?.lastName || 'N/A';

              const contact = user?.profileDetails?.contact || 'N/A';

              return (

                <tr key={index}>

                  <td>{`${firstName} ${lastName}`}</td>

                  <td>{user?.email || 'N/A'}</td>

                  <td>{contact}</td>

                  <td><button className='delete-btn' onClick={() => handleDelete(user.\_id)}>Delete</button></td>

                </tr>

              );

            })}

          </tbody>

        </table>

      );

    }

    return (

      <table>

        <thead>

          <tr>

            <th>Company Name</th>

            <th>Email</th>

            <th>Contact</th>

            <th></th>

          </tr>

        </thead>

        <tbody>

          {usersData.map((user, index) => {

            const companyName = user?.profileDetails?.companyName || 'N/A';

            const contact = user?.profileDetails?.contact || 'N/A';

            return (

              <tr key={index}>

                <td>{companyName}</td>

                <td>{user?.email || 'N/A'}</td>

                <td>{contact}</td>

                <td><button className='delete-btn'>Delete</button></td>

              </tr>

            );

          })}

        </tbody>

      </table>

    );

  };

  return (

    <div className="usersadminsection">

      <div className="leftusersadminsection">

      <h2>Users</h2>

          <div className="dropdownad-container">

          <label className="dropdownad-label">Select User Type:</label>

          <select className="user-type-dropdown" value={selectedType} onChange={handleDropdownChange}>

            <option value="Students">Students</option>

            <option value="Employers">Employers</option>

          </select>

        </div>

        <div className="userslistcontainer">{renderTable()}</div>

      </div>

      <div className="rightusersadminsection">

        {/\* Render Pie chart \*/}

          <div className="chart-container">

           <Pie data={chartData} options={{ responsive: true, plugins: { legend: { display: true } } }} />

       </div>

      </div>

    </div>

  );

};

// Tags Section Component

const TagChart = () => {

  const [tagData, setTagData] = useState([]);

  const [loading, setLoading] = useState(true);

  const [selectedTag, setSelectedTag] = useState(null);

  const [students, setStudents] = useState([]);

  const [studentsLoading, setStudentsLoading] = useState(false);

  useEffect(() => {

    const fetchTagData = async () => {

      try {

        const response = await axiosInstance.get('/api/student-tags');

        if (response.data.success) {

          setTagData(response.data.tags);

        }

      } catch (error) {

        console.error('Error fetching tag data:', error);

      } finally {

        setLoading(false);

      }

    };

    fetchTagData();

  }, []);

  const handleBarClick = async (event, elements) => {

    console.log('Bar clicked', elements); // Debugging

    if (elements.length > 0) {

      const index = elements[0].index; // Index of the clicked bar

      const tag = tagData[index].\_id; // Get tag from clicked bar

      console.log('Selected tag:', tag); // Debugging

      setSelectedTag(tag);

      setStudentsLoading(true);

      try {

        const response = await axiosInstance.get('/api/students-by-tag', { params: { tag } });

        if (response.data.success) {

          console.log('API response for students:', response.data); // Debugging

          setStudents(response.data.students);

        }

      } catch (error) {

        console.error('Error fetching students:', error);

      } finally {

        setStudentsLoading(false);

      }

    }

  };

  const chartData = {

    labels: tagData.map((tag) => tag.\_id),

    datasets: [{

      label: 'Number of Students Excelling',

      data: tagData.map((tag) => tag.count),

      backgroundColor: '#e0b8b8',

      borderColor: 'rgba(75, 192, 192, 1)',

      borderWidth: 0,

      barThickness: 50,

      maxBarThickness: 100,

    }],

  };

  return (

    <div className="userbestsadminsection">

    <h2>Top Performing Students in Various Categories</h2>

    <div className="userbestsadmincontianer">

        {loading ? (

          <p>Loading...</p>

        ) : (

          <Bar

            data={chartData}

            options={{

              responsive: true,

              plugins: {

                legend: { display: false },

              },

              scales: {

                x: { title: { display: true, text: 'Tags' } },

                y: { title: { display: true, text: 'Number of Students' } },

              },

              onClick: handleBarClick,

            }}

          />

        )}

        {selectedTag && (

          <StudentListModal

            tag={selectedTag}

            students={students}

            onClose={() => setSelectedTag(null)}

            loading={studentsLoading}

          />

        )}

      </div>

    </div>

  );

};

export default AdminDashboard;

**Authv2.js:**

const jwt = require('jsonwebtoken');

const verifyTokenv2 = (req, res, next) => {

  const token = req.cookies?.jwt; // Extract the token from the cookie

  if (!token) {

    return res.status(401).json({ success: false, message: 'Access denied. No token provided.' });

  }

  try {

    const decoded = jwt.verify(token, process.env.JWT\_SECRET); // Replace with your secret key

    req.user = decoded;

    next();

  } catch (err) {

    res.status(400).json({ success: false, message: 'Invalid token.' });

  }

};

module.exports = verifyTokenv2;

**Authcheck.js:**

const express = require('express');

const verifyToken = require('./authv2'); // Middleware for token verification

const router = express.Router();

// Auth-check endpoint

router.get('/api/auth-check', verifyToken, (req, res) => {

  res.status(200).json({ isAuthenticated: true, user: req.user });

});

module.exports = router;

**App.jsx:**

import { ToastContainer } from 'react-toastify';

import 'bootstrap/dist/css/bootstrap.min.css';

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

import StudentSignup from './components/pages/Signup/StudentSignup.jsx';

import EmployerSignup from './components/pages/Signup/EmployerSignup.jsx';

import Login from './components/pages/Login/Login';

import EmployerProfileCreation from './components/pages/Profile/EmployerProfileCreation.jsx';

import LoginRoles from './components/pages/Login/LoginRoles';

import Landingpage from './components/pages/Main/Landingpage';

import Studentlandpage from './components/pages/Main/studentlandpage';

import AdminLogin from './components/pages/admin/AdminLogin.jsx';

import AdminDashboard from './components/pages/admin/AdminDashboard.jsx'; // Admin dashboard component

import Employerlandpage from './components/pages/Main/employerlandpage';

import Homepage from './components/pages/Home/Homepage';

import StudentProfileCreation from './components/pages/Profile/StudentProfileCreation';

import Profile from './components/pages/Profile/Profilepage';

import Inboxpage from './components/pages/Inbox/Inboxpage';

import ProtectedRoute from './components/ProtectedRoute.jsx';

import Settings from './components/pages/Settings/settings';

import { GoogleOAuthProvider } from '@react-oauth/google';

import ForgotPassword from './components/pages/Login/ForgotPassword.jsx';

import ResetPassword from './components/pages/Login/ResetPassword.jsx';

import './App.css'

import UserProfile from './components/pages/Profile/UserProfile'; // Updated path

import AdminSignup from './components/pages/admin/AdminSignup.jsx';

import QuestionManager from './components/pages/admin/QuestionManager.jsx';

import ProtectedRoutev2 from './components/ProtectedRoutev2.jsx';

function App() {

  return (

    <GoogleOAuthProvider clientId="625352349873-hrob3g09um6f92jscfb672fb87cn4kvv.apps.googleusercontent.com">

    <BrowserRouter>

    {/\* Place ToastContainer here \*/}

    <ToastContainer />

    <Routes>

          <Route path='/' element={<Landingpage />} />

          <Route path="/profile/:id" element={<UserProfile />} />

          {/\* Protected Routes \*/}

          <Route

            path='/homepage'

            element={

              <ProtectedRoute>

                <Homepage />

              </ProtectedRoute>

            }

          />

          <Route

            path='/Inboxpage'

            element={

              <ProtectedRoute>

                <Inboxpage />

              </ProtectedRoute>

            }

          />

          <Route

            path='/Profile'

            element={

              <ProtectedRoute>

                <Profile />

              </ProtectedRoute>

            }

          />

          <Route

            path='/Settings'

            element={

              <ProtectedRoute>

                <Settings />

              </ProtectedRoute>

            }

          />

          <Route path="/forgot-password" element={<ForgotPassword />} />

          <Route path="/reset-password/:token" element={<ResetPassword />} />

          <Route path='/employerlandpage' element={<Employerlandpage />} />

          <Route path='/studentlandpage' element={<Studentlandpage />} />

          <Route path='/studentsignup' element={<StudentSignup />} />

          <Route path='/employersignup' element={<EmployerSignup />} />

          <Route path='/login' element={<Login />} />

          <Route path='/loginroles' element={<LoginRoles />} />

          <Route path='/studentprofilecreation' element={<StudentProfileCreation />} />

          <Route path='/employerprofilecreation' element={<EmployerProfileCreation />} />

          <Route path='/adminlogin' element={<AdminLogin />} />

          <Route path='/admindashboard' element={<ProtectedRoutev2><AdminDashboard /></ProtectedRoutev2>} />

          <Route path='/adminsignup' element={<AdminSignup />} />

          <Route path='/questionmanager' element={<QuestionManager />} />

        </Routes>

      </BrowserRouter>

    </GoogleOAuthProvider>

  );

}

export default App;