import { useState, useEffect } from "react";

import axios from "axios";

import {useNavigate} from "react-router-dom";

import AttendanceTable from "./AttendanceTable";

import jsPDF from "jspdf";

import "jspdf-autotable";

import emailjs from 'emailjs-com';

const AttendancePage = () => {

  const [selectedClass, setSelectedClass] = useState("");

  const [date, setDate] = useState("");

  const [selectedHour, setSelectedHour] = useState("");

  const [students, setStudents] = useState([]); // Store fetched students

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

  useEffect(() => {

    // Get current date in IST

    const now = new Date().toLocaleString("en-IN", { timeZone: "Asia/Kolkata" });

    setDate(now.split(",")[0]); // Extracting only the date part

  }, []);

  const handleProceed = async () => {

    if (!selectedClass || !selectedHour) {

      alert("Please select all fields before proceeding.");

      return;

    }

    setLoading(true); // Start loading

    try {

      const formattedClassName = `attendance\_${selectedClass.toLowerCase()}s`; // Format collection name

    const response = await axios.get(`http://localhost:5000/api/attendance/${formattedClassName}`);

      if (response.data && response.data.length > 0) {

        const sortedStudents = response.data.sort((a, b) => a.USN.localeCompare(b.USN));

        setStudents(sortedStudents); // Update state with fetched data

      } else {

        alert("No students found for this class.");

      }

    } catch (error) {

      console.error("Error fetching student data:", error);

      alert("Failed to fetch student data.");

    } finally {

      setLoading(false); // Stop loading

    }

  };

  // Handle status checkbox change

  const handleStatusChange = (index) => {

    const updatedStudents = [...students];

    updatedStudents[index].status = !updatedStudents[index].status; // Toggle status

    setStudents(updatedStudents); // Update the state

  };

  // Handle validity checkbox change

  const handleValidityChange = (index) => {

    const updatedStudents = [...students];

    updatedStudents[index].validity = !updatedStudents[index].validity; // Toggle validity

    setStudents(updatedStudents); // Update the state

  };

  const downloadPDF = () => {

    const doc = new jsPDF();

    doc.text(`Attendance Report - ${selectedClass}`, 10, 10);

    doc.text(`Date: ${date} | Hour: ${selectedHour}`, 10, 20);

    const tableData = students.map((student, index) => [

      index + 1,

      student.USN,

      student.NAME,

      student.status ? "Absent" : "Present",

      student.validity ? "Submitted" : "N/A",

    ]);

    doc.autoTable({

      head: [["#", "Name", "Roll Number", "Status", "Validity"]],

      body: tableData,

      startY: 30,

    });

    doc.save(`Attendance\_${selectedClass}\_${date}.pdf`);

  };

  const navigate = useNavigate();

//   const handleSubmitAttendance = () => {

//   // Filter students marked as absent

//   const absentStudents = students.filter(student => student.status);

//   // Extract emails of absent students

//   const absentEmails = absentStudents.map(student => student.EMAIL);

//   console.log("Absent Students' Emails:", absentEmails);

//   if (absentEmails.length === 0) {

//     alert("No absent students to process.");

//   } else {

//     alert(`Absent students' emails logged in the console.`);

//   }

// };

const handleSubmitAttendance = () => {

  // Filter students who are marked as absent

  const absentStudents = students.filter(student => student.status);

  // Get the email addresses of the absent students

  const absentEmails = absentStudents

    .map(student => student.EMAIL) // Ensure the correct field name (e.g., 'EMAIL') for email addresses

    .filter(email => email); // Filter out invalid emails (undefined, null, etc.)

  // Check if there are any absent students

  if (absentEmails.length === 0) {

    alert("No absent students to process.");

  } else {

    // Loop through each absent student's email and send the email via EmailJS

    absentEmails.forEach((email) => {

      // Extract the part before @gmail.com as the name

      const toName = email.split('@')[0]; // Split the email at '@' and take the first part (before the '@')

      const formData = {

        to\_name: toName, // Dynamic name for each absent student (before @gmail.com)

        message: "Your Ward is Absent to the class", // Static message or personalized

      };

      // Send email using EmailJS

      emailjs.send(

        "service\_f1c6ujr",  // Replace with your EmailJS service ID

        "template\_l3fx72b",  // Replace with your EmailJS template ID

        formData,

        "JToMI\_Qw10kbVbGJq"  // Replace with your EmailJS user ID

      )

      .then((response) => {

        console.log(`Email sent to: ${email}, Response:`, response);

      })

      .catch((error) => {

        console.error(`Failed to send email to ${email}:`, error);

      });

    });

    alert(`Emails sent to ${absentEmails.length} absent students.`);

  }

};

  return (

    <div>

      <h2>Attendance Page</h2>

      {/\* Class & Section Selection \*/}

      <label>Select Semester & Section:</label>

      <select value={selectedClass} onChange={(e) => setSelectedClass(e.target.value)}>

        <option value="">-- Select --</option>

        <option value="5A">5A</option>

        <option value="5B">5B</option>

      </select>

      {/\* Date (Pre-filled) \*/}

      <div>

        <label>Date:</label>

        <input type="text" value={date} readOnly />

      </div>

      {/\* Hour Selection \*/}

      <label>Select Hour:</label>

      <select value={selectedHour} onChange={(e) => setSelectedHour(e.target.value)}>

        <option value="">-- Select --</option>

        <option value="1">1</option>

        <option value="2">2</option>

        <option value="3">3</option>

        <option value="4">4</option>

        <option value="5">5</option>

        <option value="6">6</option>

        <option value="7">7</option>

        <option value="LAB">LAB</option>

      </select>

      <button onClick={handleProceed}>Proceed</button>

      {/\* Show loading spinner while fetching data \*/}

      {loading && <p>Loading students...</p>}

      {/\* Display the Attendance Table if students are fetched \*/}

      {students.length > 0 && (

        <>

        <AttendanceTable

          data={students}

          onStatusChange={handleStatusChange}

          onValidityChange={handleValidityChange}

        />

        <button onClick={downloadPDF}>Download Attendance PDF</button>

        </>

      )}

      <button onClick={handleSubmitAttendance}>Submit Attendance</button>

      <button onClick={() => navigate('/trial')}>Go Back</button>

    </div>

  );

};

export default AttendancePage;