NirnAI:Intelligent Diagnostic Workstation

```
/**
 * File Name    : study-list-table.tsx
 * Project Name : NirnAI – Intelligent Diagnostic Workstation
 * Created On   : 22/07/2025
 *
 * Purpose:
 *    This component displays the list of DICOM studies uploaded by the
authenticated user.
 *    It fetches studies from the backend, transforms raw DICOM metadata, and
renders an
 *    interactive table with advanced filtering capabilities including modality-
based
 *    filtering, date range filtering, and search options.
 *
 * Description:
 *    - Fetches studies using secure API calls with authentication.
 *    - Applies transformation logic to convert backend DICOM metadata into a
 *      frontend-friendly Study interface.
 *    - Renders the data inside a highly customizable DataTable component.
 *    - Supports multi-modality filtering using dynamically computed modality
options.
 *    - Provides date-range filtering using user inputs.
 *    - Includes upload and share functionalities, powered by UploadDICOMDialog and
 *      ShareLinkDialog modals.
 *    - Automatically refreshes the study list after uploads or share actions.
 *    - Ensures user authentication using AuthRedirect before rendering content.
 *
 * Author(s)    : Kiran Raj R
 * Version      : 1.0
 *
 * Revisions:
 *    - 1.0 (22/07/2025): Initial version with study fetching, filtering UI, upload
dialog,
 *                        share link dialog, and integrated DataTable.
 *
 * Remarks:
 *    Nil
 */
import React, { useEffect, useState, useCallback, useMemo } from "react";
import { fetchWithAuth } from "@/utils/fetchWithAuth";
import { transformDicomData } from "@/utils/transformDicomData";
import { DataTable } from "./components/table/data-table";
import { columns } from "./components/table/columns";
import type { Study } from "./types/study";
import { Button } from "@/components/ui/button";
import { UploadDICOMDialog } from "./components/modals/upload-dicom-modal";
import { ShareLinkDialog } from "./components/modals/share-link-modal";
import { Link2, UploadCloud } from "lucide-react";
import { useUserStore } from "@/store/user-store"; // Import the user store
import { AuthRedirect } from "@/components/auth/auth-redirect";
// Create this if needed
```

```tsx
const Studies: React.FC = () => {
 const [data, setData] = useState<Study[]>([]);
 const [filteredData, setFilteredData] = useState<Study[]>([]);
 const [loading, setLoading] = useState(false);
 const [showUpload, setShowUpload] = useState(false);
 const [showShare, setShowShare] = useState(false);
 const [shareToken, setShareToken] = useState("");
 const [selectedModalities, setSelectedModalities] = useState<string[]>([]);
 const [startDate, setStartDate] = useState<string>("");
 const [endDate, setEndDate] = useState<string>("");
 const user = useUserStore((state) => state.user);

 const modalityOptions = useMemo(() => {
  const modalities = new Set<string>();
  data.forEach((study) => {
   if (study.modalities) {
    study.modalities
     .split(",")
     .forEach((mod) => modalities.add(mod.trim()));
   }
  });
  return Array.from(modalities).sort();
 }, [data]);

 const parseStudyDate = (study: Study): Date => {
  if (!study.studyDate) return new Date(0);
  if (study.studyDate.includes("/")) {
   const [day, month, year] = study.studyDate.split("/").map(Number);
   return new Date(year, month - 1, day);
  }
  return new Date(study.studyDate);
 };

 // Combined filter function
 const filterStudies = useCallback(() => {
  let filtered = [...data];

  // Apply modality filter
  if (selectedModalities.length > 0) {
   filtered = filtered.filter((study) => {
    if (!study.modalities) return false;
    const studyModalities = study.modalities
     .split(",")
     .map((m) => m.trim());
    return selectedModalities.some((mod) => studyModalities.includes(mod));
   });
  }
```

```
 // Apply date range filter
 if (startDate || endDate) {
  const start = startDate ? new Date(startDate) : null;
  const end = endDate ? new Date(endDate) : null;

  filtered = filtered.filter((study) => {
   const studyDate = parseStudyDate(study);

   if (start && studyDate < start) return false;
   if (end && studyDate > end) return false;
   return true;
  });
 }

 setFilteredData(filtered);
}, [data, selectedModalities, startDate, endDate]);

// Apply filters when any filter changes
useEffect(() => {
 filterStudies();
}, [filterStudies]);

const fetchStudies = useCallback(async () => {
 try {
  setLoading(true);
  const userId = user?.id || sessionStorage.getItem("userId");
  if (!userId) throw new Error("User ID not found");

  const params = new URLSearchParams({ user_id: userId });
  const url = `${
   import.meta.env.VITE_API_URL
  }/api/getStudies/studies?${params}`;

  const res = await fetchWithAuth(url);
  if (!res.ok) throw new Error("Failed to fetch");

  const rawData = await res.json();
  const transformed = transformDicomData(rawData);
  setData(transformed);
  setFilteredData(transformed);
 } catch (err) {
  console.error("Error loading studies:", err);
 } finally {
  setLoading(false);
 }
}, [user?.id]);
```

```
 const fetchShareLink = async () => {
  try {
   const userId = user?.id || sessionStorage.getItem("userId");
   if (!userId) throw new Error("User ID not found");
   const res = await fetch(`${import.meta.env.VITE_API_URL}/api/getToken`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ userId }),
   });
   if (!res.ok) throw new Error("Failed to fetch share link");
   return await res.json().then((data) => data.token);
  } catch (err) {
   console.error("Error fetching share link:", err);
   return "";
  }
 };

 useEffect(() => {
  fetchStudies();
 }, [fetchStudies]);

 return (
  <div className="p-2 sm:p-3">
   <AuthRedirect />
   <div className="flex flex-col sm:flex-row justify-between items-start sm:items-
center mb-4 gap-4">
    <h2 className="text-xl sm:text-2xl font-bold text-primary-dark p-2 sm:p-4">
     Study List
    </h2>
    <div className="flex flex-col sm:flex-row gap-2 w-full sm:w-auto">
     <Button
      className="bg-primary-dark w-full sm:w-48 text-white text-sm"
      onClick={() => setShowUpload(true)}
     >
      <UploadCloud className="mr-2 h-4 w-4" /> Upload DICOM
     </Button>
     <UploadDICOMDialog
      show={showUpload}
      onClose={setShowUpload}
      onUploadComplete={() => {
       fetchStudies(); // refresh data
       // setShowUpload(false); // optionally close dialog
      }}
     />

     <Button
      className="bg-primary-dark w-full sm:w-48 text-white text-sm"
      onClick={async () => {
```

```
        setShowShare(true);
        const token = await fetchShareLink();
        setShareToken(token);
       }}
      >
       <Link2 className="mr-2 h-4 w-4" /> Share Link
      </Button>
      <ShareLinkDialog
       url={shareToken}
       open={showShare}
       onOpenChange={setShowShare}
      />
     </div>
    </div>

    <div className="w-full lg:w-[75%] mx-auto">
     <DataTable
      columns={columns}
      data={filteredData}
      loading={loading}
      modalityOptions={modalityOptions.map((mod) => ({
       value: mod,
       label: mod,
      }))}
      onModalityFilterChange={setSelectedModalities}
      startDate={startDate}
      endDate={endDate}
      onStartDateChange={setStartDate}
      onEndDateChange={setEndDate}
      onShareComplete={fetchStudies}
     />
    </div>
   </div>
  );
};

export default Studies;
```