import React, { useEffect, useState, useRef, useCallback } from "react";

import { createTheme } from "@mui/material/styles";

import { supabase } from "./supabaseClient";

import PharmacistViewUI from "./PharmacistViewUI";

import AddOrderQuantityDialog from "./Pharmacist/AddOrderQuantityDialog";

import AddMedicineDialog from "./Pharmacist/AddMedicineDialog";

import StatisticsBoard from "./StatisticsBoard";

import PharmaInventory from "./PharmaInventory";

import CharacterModal from "./CharacterModal";

interface Thuoc {

id: string;

ten\_thuoc: string;

don\_vi: string;

so\_luong\_ton: number;

gia\_nhap: number;

gia\_ban: number;

so\_luong\_dat\_hang: number;

duong\_dung: string;

}

const defaultTheme = (mode: "light" | "dark") =>

createTheme({

palette: { mode },

shape: { borderRadius: 12 },

});

const capitalizeFirstLetter = (str: string) =>

str.toLowerCase().replace(/(?:^|\s)\w/g, (match) => match.toUpperCase());

const PharmacistView: React.FC = () => {

const [mode, setMode] = useState<"light" | "dark">("light");

const theme = defaultTheme(mode);

const [drawerOpen, setDrawerOpen] = useState(true);

const [thuocList, setThuocList] = useState<Thuoc[]>([]);

const [openAddDialog, setOpenAddDialog] = useState(false);

const [successMessage, setSuccessMessage] = useState("");

const [newThuoc, setNewThuoc] = useState<Thuoc>({

id: "",

ten\_thuoc: "",

don\_vi: "",

so\_luong\_ton: 0,

gia\_nhap: 0,

gia\_ban: 0,

so\_luong\_dat\_hang: 0,

duong\_dung: "",

});

const tenThuocRef = useRef<HTMLInputElement>(null);

const donViRef = useRef<HTMLInputElement>(null);

const soLuongTonRef = useRef<HTMLInputElement>(null);

const giaNhapRef = useRef<HTMLInputElement>(null);

const giaBanRef = useRef<HTMLInputElement>(null);

const duongDungRef = useRef<HTMLInputElement>(null);

const [openOrderDialog, setOpenOrderDialog] = useState(false);

const [selectedThuoc, setSelectedThuoc] = useState<Thuoc | null>(null);

const [showPharmaInventory, setShowPharmaInventory] = useState(false);

const fetchThuocList = useCallback(async () => {

const { data, error } = await supabase.from("thuoc").select("\*");

if (!error && data) {

setThuocList(data as Thuoc[]);

}

}, []);

useEffect(() => {

fetchThuocList();

}, [fetchThuocList]);

const handleChange = (e: React.ChangeEvent<HTMLInputElement>) => {

const { name, value } = e.target;

setNewThuoc((prev) => ({

...prev,

[name]: [

"so\_luong\_ton",

"gia\_nhap",

"gia\_ban",

"so\_luong\_dat\_hang",

].includes(name)

? Number(value)

: value,

}));

};

const handleKeyDown = (

e: React.KeyboardEvent,

nextRef: React.RefObject<HTMLInputElement> | null

) => {

if (e.key === "Enter") {

if (nextRef && nextRef.current) nextRef.current.focus();

else handleAddThuoc();

}

};

const handleAddThuoc = async () => {

const formattedTenThuoc = capitalizeFirstLetter(newThuoc.ten\_thuoc.trim());

if (!formattedTenThuoc) {

alert("Tên thuốc không được để trống");

return;

}

// Kiểm tra xem đã có thuốc đó chưa

const { data: existing, error: checkError } = await supabase

.from("thuoc")

.select("\*")

.eq("ten\_thuoc", formattedTenThuoc)

.single();

if (checkError && checkError.code !== "PGRST116") {

alert("Lỗi khi kiểm tra thuốc: " + checkError.message);

return;

}

if (existing) {

// Nếu đã tồn tại, cập nhật số lượng tồn, giá nhập, giá bán, số lượng đặt và đường dùng

const updatedSoLuongDatHang =

newThuoc.so\_luong\_dat\_hang > 0

? newThuoc.so\_luong\_dat\_hang

: existing.so\_luong\_dat\_hang;

const updatedThuoc: Thuoc = {

...existing,

so\_luong\_ton: existing.so\_luong\_ton + newThuoc.so\_luong\_ton,

gia\_nhap: newThuoc.gia\_nhap,

gia\_ban: newThuoc.gia\_ban,

so\_luong\_dat\_hang: updatedSoLuongDatHang,

duong\_dung: newThuoc.duong\_dung || existing.duong\_dung,

};

const { error: updateError } = await supabase

.from("thuoc")

.update({

so\_luong\_ton: updatedThuoc.so\_luong\_ton,

gia\_nhap: updatedThuoc.gia\_nhap,

gia\_ban: updatedThuoc.gia\_ban,

so\_luong\_dat\_hang: updatedThuoc.so\_luong\_dat\_hang,

duong\_dung: updatedThuoc.duong\_dung,

})

.eq("id", existing.id);

if (updateError) {

alert("Lỗi khi cập nhật thuốc: " + updateError.message);

return;

}

setThuocList((prev) =>

prev.map((t) => (t.id === existing.id ? updatedThuoc : t))

);

setSuccessMessage("Đã cập nhật thuốc thành công!");

} else {

// Nếu chưa có, thêm mới vào bảng

const { data: inserted, error: insertError } = await supabase

.from("thuoc")

.insert([

{

ten\_thuoc: formattedTenThuoc,

don\_vi: newThuoc.don\_vi,

so\_luong\_ton: newThuoc.so\_luong\_ton,

gia\_nhap: newThuoc.gia\_nhap,

gia\_ban: newThuoc.gia\_ban,

so\_luong\_dat\_hang: newThuoc.so\_luong\_dat\_hang,

duong\_dung: newThuoc.duong\_dung,

},

])

.select();

if (insertError) {

alert("Lỗi khi thêm thuốc: " + insertError.message);

return;

}

setThuocList((prev) => [...prev, ...(inserted as Thuoc[])]);

setSuccessMessage("Đã thêm thuốc mới thành công!");

}

// Reset lại form

setNewThuoc({

id: "",

ten\_thuoc: "",

don\_vi: "",

so\_luong\_ton: 0,

gia\_nhap: 0,

gia\_ban: 0,

so\_luong\_dat\_hang: 0,

duong\_dung: "",

});

setOpenAddDialog(false);

};

// Hàm cập nhật số lượng tồn cho thuốc cũ

const handleUpdateStock = async (

medicineId: string,

additionalQuantity: number

) => {

try {

// Lấy thông tin thuốc hiện tại

const { data: currentMedicine, error: fetchError } = await supabase

.from("thuoc")

.select("\*")

.eq("id", medicineId)

.single();

if (fetchError) {

alert("Lỗi khi lấy thông tin thuốc: " + fetchError.message);

return;

}

// Cập nhật số lượng tồn

const newQuantity = currentMedicine.so\_luong\_ton + additionalQuantity;

const { error: updateError } = await supabase

.from("thuoc")

.update({ so\_luong\_ton: newQuantity })

.eq("id", medicineId);

if (updateError) {

alert("Lỗi khi cập nhật số lượng: " + updateError.message);

return;

}

// Cập nhật state local

setThuocList((prev) =>

prev.map((t) =>

t.id === medicineId ? { ...t, so\_luong\_ton: newQuantity } : t

)

);

setSuccessMessage(

`Đã nhập thêm ${additionalQuantity} ${currentMedicine.don\_vi} cho ${currentMedicine.ten\_thuoc}!`

);

} catch (error) {

alert("Có lỗi xảy ra khi cập nhật số lượng thuốc");

console.error("Error updating stock:", error);

}

};

// Hàm mới: Cập nhật thông tin thuốc (giá nhập, giá bán, số lượng đặt hàng)

const handleUpdateMedicine = async (

medicineId: string,

updatedData: Partial<Thuoc>

) => {

try {

const { error: updateError } = await supabase

.from("thuoc")

.update(updatedData)

.eq("id", medicineId);

if (updateError) {

alert("Lỗi khi cập nhật thông tin thuốc: " + updateError.message);

return;

}

// Cập nhật state local

setThuocList((prev) =>

prev.map((t) => (t.id === medicineId ? { ...t, ...updatedData } : t))

);

// Tạo thông báo thành công chi tiết

const updateMessages = [];

if (updatedData.gia\_nhap !== undefined) updateMessages.push("giá nhập");

if (updatedData.gia\_ban !== undefined) updateMessages.push("giá bán");

if (updatedData.so\_luong\_dat\_hang !== undefined)

updateMessages.push("số lượng đặt hàng");

setSuccessMessage(`Đã cập nhật ${updateMessages.join(", ")} thành công!`);

// Tự động refresh danh sách sau một khoảng thời gian ngắn

setTimeout(() => {

fetchThuocList();

}, 500);

} catch (error) {

alert("Có lỗi xảy ra khi cập nhật thông tin thuốc");

console.error("Error updating medicine:", error);

}

};

const handleOpenOrderDialog = (thuoc: Thuoc) => {

setSelectedThuoc(thuoc);

setOpenOrderDialog(true);

};

const handleCloseOrderDialog = () => {

setOpenOrderDialog(false);

setSelectedThuoc(null);

};

const handleSaveOrderQuantity = async (quantity: number) => {

if (!selectedThuoc) return;

const { error } = await supabase

.from("thuoc")

.update({ so\_luong\_dat\_hang: quantity })

.eq("id", selectedThuoc.id);

if (error) {

alert("Lỗi cập nhật số lượng đặt hàng: " + error.message);

return;

}

setThuocList((prev) =>

prev.map((t) =>

t.id === selectedThuoc.id ? { ...t, so\_luong\_dat\_hang: quantity } : t

)

);

setSuccessMessage("Cập nhật số lượng đặt hàng thành công!");

handleCloseOrderDialog();

};

const getRowStyle = (thuoc: Thuoc) => ({

backgroundColor:

thuoc.so\_luong\_ton <= thuoc.so\_luong\_dat\_hang ? "orange" : "transparent",

});

// Cấu hình columns với căn phải các cột số, căn trái cột tên thuốc

const columns = [

{

field: "ten\_thuoc",

headerName: "Tên thuốc",

flex: 1,

headerAlign: "left",

align: "left",

},

{

field: "don\_vi",

headerName: "Đơn vị",

width: 100,

headerAlign: "right",

align: "right",

},

{

field: "so\_luong\_ton",

headerName: "Tồn kho",

width: 100,

headerAlign: "right",

align: "right",

},

{

field: "gia\_nhap",

headerName: "Giá nhập",

width: 100,

headerAlign: "right",

align: "right",

renderCell: (params: any) => <span>{params.value.toLocaleString()}</span>,

},

{

field: "gia\_ban",

headerName: "Giá bán",

width: 100,

headerAlign: "right",

align: "right",

renderCell: (params: any) => <span>{params.value.toLocaleString()}</span>,

},

{

field: "so\_luong\_dat\_hang",

headerName: "Số lượng đặt hàng",

width: 180,

headerAlign: "right",

align: "right",

renderCell: (params: any) => (

<button

onClick={() => handleOpenOrderDialog(params.row)}

style={{ cursor: "pointer" }}

>

{params.value}

</button>

),

},

{

field: "duong\_dung",

headerName: "Đường dùng",

width: 150,

headerAlign: "left",

align: "left",

},

];

// Sắp xếp danh sách thuốc theo tên alphabet trước khi truyền xuống UI

const sortedThuocList = [...thuocList].sort((a, b) =>

a.ten\_thuoc.localeCompare(b.ten\_thuoc)

);

const handleLogout = async () => {

await supabase.auth.signOut();

window.location.reload();

};

// MỞ PHARMAINVENTORY KHI CLICK LINH VẬT

const handleCharacterClick = () => {

setShowPharmaInventory(true);

};

const handleClosePharmaInventory = () => {

setShowPharmaInventory(false);

};

const statisticsData = [

{

label: "Tổng doanh thu",

value: thuocList.reduce(

(total, t) => total + t.so\_luong\_ton \* t.gia\_ban,

0

),

},

{ label: "Tổng số thuốc", value: thuocList.length },

{

label: "Tổng số lượng đặt hàng",

value: thuocList.reduce((total, t) => total + t.so\_luong\_dat\_hang, 0),

},

];

return (

<>

<PharmacistViewUI

thuocList={sortedThuocList}

columns={columns}

mode={mode}

setMode={setMode}

theme={theme}

openDialog={openAddDialog}

setOpenDialog={setOpenAddDialog}

newThuoc={newThuoc}

handleChange={handleChange}

handleAddThuoc={handleAddThuoc}

handleKeyDown={handleKeyDown}

tenThuocRef={tenThuocRef}

donViRef={donViRef}

soLuongTonRef={soLuongTonRef}

giaNhapRef={giaNhapRef}

giaBanRef={giaBanRef}

duongDungRef={duongDungRef}

successMessage={successMessage}

setSuccessMessage={setSuccessMessage}

getRowStyle={getRowStyle}

handleLogout={handleLogout}

drawerOpen={drawerOpen}

setDrawerOpen={setDrawerOpen}

statisticsData={statisticsData}

/>

<CharacterModal

onClose={() => {}}

onCharacterClick={handleCharacterClick}

/>

{showPharmaInventory && (

<PharmaInventory onClose={handleClosePharmaInventory} />

)}

{selectedThuoc && (

<AddOrderQuantityDialog

open={openOrderDialog}

onClose={handleCloseOrderDialog}

thuoc={selectedThuoc}

onSave={handleSaveOrderQuantity}

/>

)}

<AddMedicineDialog

openDialog={openAddDialog}

setOpenDialog={setOpenAddDialog}

newThuoc={newThuoc}

handleChange={handleChange}

handleAddThuoc={handleAddThuoc}

handleKeyDown={handleKeyDown}

tenThuocRef={tenThuocRef}

donViRef={donViRef}

soLuongTonRef={soLuongTonRef}

giaNhapRef={giaNhapRef}

giaBanRef={giaBanRef}

duongDungRef={duongDungRef}

// Props mới cho chức năng thuốc cũ với khả năng chỉnh sửa

availableMedicines={sortedThuocList}

handleUpdateStock={handleUpdateStock}

handleUpdateMedicine={handleUpdateMedicine}

/>

</>

);

};

export default PharmacistView;