/// src/Pharmacist/AddMedicineDialog.tsx

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

import {

Dialog,

DialogTitle,

DialogContent,

DialogActions,

Button,

TextField,

Box,

Tabs,

Tab,

Autocomplete,

Typography,

Paper,

Divider,

} from "@mui/material";

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;

}

interface AddMedicineDialogProps {

openDialog: boolean;

setOpenDialog: React.Dispatch<React.SetStateAction<boolean>>;

newThuoc: Thuoc;

handleChange: (e: React.ChangeEvent<HTMLInputElement>) => void;

handleAddThuoc: () => Promise<void>;

handleKeyDown: (

e: React.KeyboardEvent,

nextRef: React.RefObject<HTMLInputElement> | null

) => void;

tenThuocRef: React.RefObject<HTMLInputElement>;

donViRef: React.RefObject<HTMLInputElement>;

soLuongTonRef: React.RefObject<HTMLInputElement>;

giaNhapRef: React.RefObject<HTMLInputElement>;

giaBanRef: React.RefObject<HTMLInputElement>;

duongDungRef: React.RefObject<HTMLInputElement>;

// Props mới cho chức năng thuốc cũ

availableMedicines?: Thuoc[];

handleUpdateStock?: (

medicineId: string,

additionalQuantity: number

) => Promise<void>;

handleUpdateMedicine?: (

medicineId: string,

updatedData: Partial<Thuoc>

) => Promise<void>;

}

const AddMedicineDialog: React.FC<AddMedicineDialogProps> = ({

openDialog,

setOpenDialog,

newThuoc,

handleChange,

handleAddThuoc,

handleKeyDown,

tenThuocRef,

donViRef,

soLuongTonRef,

giaNhapRef,

giaBanRef,

duongDungRef,

availableMedicines = [],

handleUpdateStock,

handleUpdateMedicine,

}) => {

const [activeTab, setActiveTab] = useState(0); // 0: thuốc mới, 1: thuốc cũ

const [selectedMedicine, setSelectedMedicine] = useState<Thuoc | null>(null);

const [additionalQuantity, setAdditionalQuantity] = useState<number>(0);

const [autocompleteValue, setAutocompleteValue] = useState<string>("");

// State cho việc chỉnh sửa thông tin thuốc cũ

const [editingData, setEditingData] = useState({

gia\_nhap: 0,

gia\_ban: 0,

so\_luong\_dat\_hang: 0,

});

// Reset editing data khi chọn thuốc mới

useEffect(() => {

if (selectedMedicine) {

setEditingData({

gia\_nhap: selectedMedicine.gia\_nhap,

gia\_ban: selectedMedicine.gia\_ban,

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

});

}

}, [selectedMedicine]);

const handleClose = () => {

setOpenDialog(false);

// Reset state khi đóng dialog

setActiveTab(0);

setSelectedMedicine(null);

setAdditionalQuantity(0);

setAutocompleteValue("");

setEditingData({ gia\_nhap: 0, gia\_ban: 0, so\_luong\_dat\_hang: 0 });

};

const handleTabChange = (event: React.SyntheticEvent, newValue: number) => {

setActiveTab(newValue);

// Reset state khi chuyển tab

setSelectedMedicine(null);

setAdditionalQuantity(0);

setAutocompleteValue("");

setEditingData({ gia\_nhap: 0, gia\_ban: 0, so\_luong\_dat\_hang: 0 });

};

const handleMedicineSelect = (event: any, value: Thuoc | null) => {

setSelectedMedicine(value);

setAutocompleteValue(value ? value.ten\_thuoc : "");

if (value) {

setEditingData({

gia\_nhap: value.gia\_nhap,

gia\_ban: value.gia\_ban,

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

});

}

};

const handleAdditionalQuantityChange = (

e: React.ChangeEvent<HTMLInputElement>

) => {

setAdditionalQuantity(Number(e.target.value));

};

const handleEditingDataChange =

(field: keyof typeof editingData) =>

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

setEditingData((prev) => ({

...prev,

[field]: Number(e.target.value),

}));

};

const handleSubmit = async () => {

if (activeTab === 0) {

// Chế độ thuốc mới

await handleAddThuoc();

} else {

// Chế độ thuốc cũ

if (selectedMedicine && handleUpdateStock && handleUpdateMedicine) {

try {

// Cập nhật số lượng tồn nếu có nhập thêm

if (additionalQuantity > 0) {

await handleUpdateStock(selectedMedicine.id, additionalQuantity);

}

// Kiểm tra xem có thay đổi thông tin nào không

const hasChanges =

editingData.gia\_nhap !== selectedMedicine.gia\_nhap ||

editingData.gia\_ban !== selectedMedicine.gia\_ban ||

editingData.so\_luong\_dat\_hang !==

selectedMedicine.so\_luong\_dat\_hang;

// Cập nhật thông tin khác nếu có thay đổi

if (hasChanges) {

await handleUpdateMedicine(selectedMedicine.id, editingData);

}

handleClose();

} catch (error) {

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

}

}

}

};

const isSubmitDisabled = () => {

if (activeTab === 0) {

// Validation cho thuốc mới

return !newThuoc.ten\_thuoc || !newThuoc.don\_vi || !newThuoc.duong\_dung;

} else {

// Validation cho thuốc cũ - cho phép chỉ cập nhật thông tin mà không cần nhập thêm số lượng

return !selectedMedicine;

}

};

const hasChanges =

selectedMedicine &&

(editingData.gia\_nhap !== selectedMedicine.gia\_nhap ||

editingData.gia\_ban !== selectedMedicine.gia\_ban ||

editingData.so\_luong\_dat\_hang !== selectedMedicine.so\_luong\_dat\_hang ||

additionalQuantity > 0);

return (

<Dialog open={openDialog} onClose={handleClose} maxWidth="sm" fullWidth>

<DialogTitle>

<Typography variant="h6" component="div" sx={{ mb: 2 }}>

Nhập thông tin thuốc

</Typography>

<Tabs

value={activeTab}

onChange={handleTabChange}

aria-label="medicine tabs"

>

<Tab label="Thuốc mới" />

<Tab label="Thuốc cũ" />

</Tabs>

</DialogTitle>

<DialogContent>

{activeTab === 0 ? (

// Form cho thuốc mới (đầy đủ thông tin)

<Box

component="form"

sx={{

display: "flex",

flexDirection: "column",

gap: 2,

mt: 1,

}}

>

<TextField

label="Tên thuốc"

name="ten\_thuoc"

value={newThuoc.ten\_thuoc}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, donViRef)}

inputRef={tenThuocRef}

fullWidth

autoFocus

required

/>

<TextField

label="Đơn vị"

name="don\_vi"

value={newThuoc.don\_vi}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, soLuongTonRef)}

inputRef={donViRef}

fullWidth

required

/>

<TextField

label="Số lượng tồn"

name="so\_luong\_ton"

type="number"

value={newThuoc.so\_luong\_ton}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, giaNhapRef)}

inputRef={soLuongTonRef}

fullWidth

required

/>

<TextField

label="Giá nhập (VND)"

name="gia\_nhap"

type="number"

value={newThuoc.gia\_nhap}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, giaBanRef)}

inputRef={giaNhapRef}

fullWidth

required

/>

<TextField

label="Giá bán (VND)"

name="gia\_ban"

type="number"

value={newThuoc.gia\_ban}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, duongDungRef)}

inputRef={giaBanRef}

fullWidth

required

/>

<TextField

label="Số lượng đặt hàng"

name="so\_luong\_dat\_hang"

type="number"

value={newThuoc.so\_luong\_dat\_hang}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, duongDungRef)}

fullWidth

/>

<TextField

label="Đường dùng"

name="duong\_dung"

value={newThuoc.duong\_dung}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, null)}

inputRef={duongDungRef}

helperText="Ví dụ: Uống, Tiêm, Bôi, Nhỏ mắt…"

fullWidth

required

/>

</Box>

) : (

// Form cho thuốc cũ (dropdown + chỉnh sửa thông tin)

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: 2,

mt: 1,

}}

>

<Autocomplete

options={availableMedicines}

getOptionLabel={(option) => option.ten\_thuoc}

value={selectedMedicine}

onChange={handleMedicineSelect}

inputValue={autocompleteValue}

onInputChange={(event, newInputValue) => {

setAutocompleteValue(newInputValue);

}}

renderInput={(params) => (

<TextField

{...params}

label="Tìm thuốc có sẵn"

placeholder="Nhập tên thuốc để tìm kiếm..."

required

autoFocus

/>

)}

renderOption={(props, option) => (

<li {...props}>

<Box>

<Typography variant="body1" sx={{ fontWeight: 500 }}>

{option.ten\_thuoc}

</Typography>

<Typography variant="body2" color="text.secondary">

Đơn vị: {option.don\_vi} | Tồn kho: {option.so\_luong\_ton}

</Typography>

<Typography variant="caption" color="text.secondary">

Giá bán: {option.gia\_ban.toLocaleString()} VND

</Typography>

</Box>

</li>

)}

noOptionsText="Không tìm thấy thuốc nào"

filterOptions={(options, { inputValue }) =>

options.filter((option) =>

option.ten\_thuoc

.toLowerCase()

.includes(inputValue.toLowerCase())

)

}

/>

{selectedMedicine && (

<>

<Paper

elevation={1}

sx={{

p: 2,

backgroundColor: "grey.50",

borderRadius: 1,

}}

>

<Typography

variant="subtitle2"

sx={{ fontWeight: 600, mb: 1 }}

>

Thuốc đã chọn: {selectedMedicine.ten\_thuoc}

</Typography>

<Typography variant="body2" color="text.secondary">

Đơn vị: {selectedMedicine.don\_vi}

</Typography>

<Typography variant="body2" color="text.secondary">

Số lượng hiện tại: {selectedMedicine.so\_luong\_ton}{" "}

{selectedMedicine.don\_vi}

</Typography>

<Typography variant="body2" color="text.secondary">

Đường dùng: {selectedMedicine.duong\_dung}

</Typography>

</Paper>

<Divider sx={{ my: 1 }} />

<Typography variant="subtitle2" sx={{ fontWeight: 600 }}>

Nhập hàng và cập nhật thông tin

</Typography>

<TextField

label="Số lượng nhập thêm"

type="number"

value={additionalQuantity}

onChange={handleAdditionalQuantityChange}

inputProps={{ min: 0 }}

helperText="Nhập 0 nếu chỉ muốn cập nhật thông tin"

fullWidth

/>

<Box sx={{ display: "flex", gap: 2 }}>

<TextField

label="Giá nhập (VND)"

type="number"

value={editingData.gia\_nhap}

onChange={handleEditingDataChange("gia\_nhap")}

inputProps={{ min: 0 }}

fullWidth

/>

<TextField

label="Giá bán (VND)"

type="number"

value={editingData.gia\_ban}

onChange={handleEditingDataChange("gia\_ban")}

inputProps={{ min: 0 }}

fullWidth

/>

</Box>

<TextField

label="Số lượng đặt hàng"

type="number"

value={editingData.so\_luong\_dat\_hang}

onChange={handleEditingDataChange("so\_luong\_dat\_hang")}

inputProps={{ min: 0 }}

fullWidth

/>

{(additionalQuantity > 0 || hasChanges) && (

<Paper

elevation={0}

sx={{

p: 2,

backgroundColor: "info.light",

color: "info.contrastText",

borderRadius: 1,

}}

>

<Typography variant="body2" sx={{ mb: 1, fontWeight: 500 }}>

Thay đổi sẽ được áp dụng:

</Typography>

{additionalQuantity > 0 && (

<Typography variant="body2">

• Số lượng sau khi nhập:{" "}

{selectedMedicine.so\_luong\_ton + additionalQuantity}{" "}

{selectedMedicine.don\_vi}

</Typography>

)}

{editingData.gia\_nhap !== selectedMedicine.gia\_nhap && (

<Typography variant="body2">

• Giá nhập: {selectedMedicine.gia\_nhap.toLocaleString()}{" "}

→ {editingData.gia\_nhap.toLocaleString()} VND

</Typography>

)}

{editingData.gia\_ban !== selectedMedicine.gia\_ban && (

<Typography variant="body2">

• Giá bán: {selectedMedicine.gia\_ban.toLocaleString()} →{" "}

{editingData.gia\_ban.toLocaleString()} VND

</Typography>

)}

{editingData.so\_luong\_dat\_hang !==

selectedMedicine.so\_luong\_dat\_hang && (

<Typography variant="body2">

• Số lượng đặt hàng:{" "}

{selectedMedicine.so\_luong\_dat\_hang} →{" "}

{editingData.so\_luong\_dat\_hang}

</Typography>

)}

</Paper>

)}

</>

)}

</Box>

)}

</DialogContent>

<DialogActions>

<Button onClick={handleClose} color="secondary">

Hủy

</Button>

<Button

onClick={handleSubmit}

variant="contained"

color="primary"

disabled={isSubmitDisabled()}

>

{activeTab === 0 ? "Lưu" : hasChanges ? "Cập nhật" : "Đóng"}

</Button>

</DialogActions>

</Dialog>

);

};

export default AddMedicineDialog;

/ src/Pharmacist/AddMedicineDialog.tsx

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

import {

Dialog,

DialogTitle,

DialogContent,

DialogActions,

Button,

TextField,

Box,

Tabs,

Tab,

Autocomplete,

Typography,

Paper,

Divider,

} from "@mui/material";

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;

}

interface AddMedicineDialogProps {

openDialog: boolean;

setOpenDialog: React.Dispatch<React.SetStateAction<boolean>>;

newThuoc: Thuoc;

handleChange: (e: React.ChangeEvent<HTMLInputElement>) => void;

handleAddThuoc: () => Promise<void>;

handleKeyDown: (

e: React.KeyboardEvent,

nextRef: React.RefObject<HTMLInputElement> | null

) => void;

tenThuocRef: React.RefObject<HTMLInputElement>;

donViRef: React.RefObject<HTMLInputElement>;

soLuongTonRef: React.RefObject<HTMLInputElement>;

giaNhapRef: React.RefObject<HTMLInputElement>;

giaBanRef: React.RefObject<HTMLInputElement>;

duongDungRef: React.RefObject<HTMLInputElement>;

// Props mới cho chức năng thuốc cũ

availableMedicines?: Thuoc[];

handleUpdateStock?: (

medicineId: string,

additionalQuantity: number

) => Promise<void>;

handleUpdateMedicine?: (

medicineId: string,

updatedData: Partial<Thuoc>

) => Promise<void>;

}

const AddMedicineDialog: React.FC<AddMedicineDialogProps> = ({

openDialog,

setOpenDialog,

newThuoc,

handleChange,

handleAddThuoc,

handleKeyDown,

tenThuocRef,

donViRef,

soLuongTonRef,

giaNhapRef,

giaBanRef,

duongDungRef,

availableMedicines = [],

handleUpdateStock,

handleUpdateMedicine,

}) => {

const [activeTab, setActiveTab] = useState(0); // 0: thuốc mới, 1: thuốc cũ

const [selectedMedicine, setSelectedMedicine] = useState<Thuoc | null>(null);

const [additionalQuantity, setAdditionalQuantity] = useState<number>(0);

const [autocompleteValue, setAutocompleteValue] = useState<string>("");

// State cho việc chỉnh sửa thông tin thuốc cũ

const [editingData, setEditingData] = useState({

gia\_nhap: 0,

gia\_ban: 0,

so\_luong\_dat\_hang: 0,

});

// Reset editing data khi chọn thuốc mới

useEffect(() => {

if (selectedMedicine) {

setEditingData({

gia\_nhap: selectedMedicine.gia\_nhap,

gia\_ban: selectedMedicine.gia\_ban,

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

});

}

}, [selectedMedicine]);

const handleClose = () => {

setOpenDialog(false);

// Reset state khi đóng dialog

setActiveTab(0);

setSelectedMedicine(null);

setAdditionalQuantity(0);

setAutocompleteValue("");

setEditingData({ gia\_nhap: 0, gia\_ban: 0, so\_luong\_dat\_hang: 0 });

};

const handleTabChange = (event: React.SyntheticEvent, newValue: number) => {

setActiveTab(newValue);

// Reset state khi chuyển tab

setSelectedMedicine(null);

setAdditionalQuantity(0);

setAutocompleteValue("");

setEditingData({ gia\_nhap: 0, gia\_ban: 0, so\_luong\_dat\_hang: 0 });

};

const handleMedicineSelect = (event: any, value: Thuoc | null) => {

setSelectedMedicine(value);

setAutocompleteValue(value ? value.ten\_thuoc : "");

if (value) {

setEditingData({

gia\_nhap: value.gia\_nhap,

gia\_ban: value.gia\_ban,

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

});

}

};

const handleAdditionalQuantityChange = (

e: React.ChangeEvent<HTMLInputElement>

) => {

setAdditionalQuantity(Number(e.target.value));

};

const handleEditingDataChange =

(field: keyof typeof editingData) =>

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

setEditingData((prev) => ({

...prev,

[field]: Number(e.target.value),

}));

};

const handleSubmit = async () => {

if (activeTab === 0) {

// Chế độ thuốc mới

await handleAddThuoc();

} else {

// Chế độ thuốc cũ

if (selectedMedicine && handleUpdateStock && handleUpdateMedicine) {

try {

// Cập nhật số lượng tồn nếu có nhập thêm

if (additionalQuantity > 0) {

await handleUpdateStock(selectedMedicine.id, additionalQuantity);

}

// Kiểm tra xem có thay đổi thông tin nào không

const hasChanges =

editingData.gia\_nhap !== selectedMedicine.gia\_nhap ||

editingData.gia\_ban !== selectedMedicine.gia\_ban ||

editingData.so\_luong\_dat\_hang !==

selectedMedicine.so\_luong\_dat\_hang;

// Cập nhật thông tin khác nếu có thay đổi

if (hasChanges) {

await handleUpdateMedicine(selectedMedicine.id, editingData);

}

handleClose();

} catch (error) {

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

}

}

}

};

const isSubmitDisabled = () => {

if (activeTab === 0) {

// Validation cho thuốc mới

return !newThuoc.ten\_thuoc || !newThuoc.don\_vi || !newThuoc.duong\_dung;

} else {

// Validation cho thuốc cũ - cho phép chỉ cập nhật thông tin mà không cần nhập thêm số lượng

return !selectedMedicine;

}

};

const hasChanges =

selectedMedicine &&

(editingData.gia\_nhap !== selectedMedicine.gia\_nhap ||

editingData.gia\_ban !== selectedMedicine.gia\_ban ||

editingData.so\_luong\_dat\_hang !== selectedMedicine.so\_luong\_dat\_hang ||

additionalQuantity > 0);

return (

<Dialog open={openDialog} onClose={handleClose} maxWidth="sm" fullWidth>

<DialogTitle>

<Typography variant="h6" component="div" sx={{ mb: 2 }}>

Nhập thông tin thuốc

</Typography>

<Tabs

value={activeTab}

onChange={handleTabChange}

aria-label="medicine tabs"

>

<Tab label="Thuốc mới" />

<Tab label="Thuốc cũ" />

</Tabs>

</DialogTitle>

<DialogContent>

{activeTab === 0 ? (

// Form cho thuốc mới (đầy đủ thông tin)

<Box

component="form"

sx={{

display: "flex",

flexDirection: "column",

gap: 2,

mt: 1,

}}

>

<TextField

label="Tên thuốc"

name="ten\_thuoc"

value={newThuoc.ten\_thuoc}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, donViRef)}

inputRef={tenThuocRef}

fullWidth

autoFocus

required

/>

<TextField

label="Đơn vị"

name="don\_vi"

value={newThuoc.don\_vi}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, soLuongTonRef)}

inputRef={donViRef}

fullWidth

required

/>

<TextField

label="Số lượng tồn"

name="so\_luong\_ton"

type="number"

value={newThuoc.so\_luong\_ton}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, giaNhapRef)}

inputRef={soLuongTonRef}

fullWidth

required

/>

<TextField

label="Giá nhập (VND)"

name="gia\_nhap"

type="number"

value={newThuoc.gia\_nhap}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, giaBanRef)}

inputRef={giaNhapRef}

fullWidth

required

/>

<TextField

label="Giá bán (VND)"

name="gia\_ban"

type="number"

value={newThuoc.gia\_ban}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, duongDungRef)}

inputRef={giaBanRef}

fullWidth

required

/>

<TextField

label="Số lượng đặt hàng"

name="so\_luong\_dat\_hang"

type="number"

value={newThuoc.so\_luong\_dat\_hang}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, duongDungRef)}

fullWidth

/>

<TextField

label="Đường dùng"

name="duong\_dung"

value={newThuoc.duong\_dung}

onChange={handleChange}

onKeyDown={(e) => handleKeyDown(e, null)}

inputRef={duongDungRef}

helperText="Ví dụ: Uống, Tiêm, Bôi, Nhỏ mắt…"

fullWidth

required

/>

</Box>

) : (

// Form cho thuốc cũ (dropdown + chỉnh sửa thông tin)

<Box

sx={{

display: "flex",

flexDirection: "column",

gap: 2,

mt: 1,

}}

>

<Autocomplete

options={availableMedicines}

getOptionLabel={(option) => option.ten\_thuoc}

value={selectedMedicine}

onChange={handleMedicineSelect}

inputValue={autocompleteValue}

onInputChange={(event, newInputValue) => {

setAutocompleteValue(newInputValue);

}}

renderInput={(params) => (

<TextField

{...params}

label="Tìm thuốc có sẵn"

placeholder="Nhập tên thuốc để tìm kiếm..."

required

autoFocus

/>

)}

renderOption={(props, option) => (

<li {...props}>

<Box>

<Typography variant="body1" sx={{ fontWeight: 500 }}>

{option.ten\_thuoc}

</Typography>

<Typography variant="body2" color="text.secondary">

Đơn vị: {option.don\_vi} | Tồn kho: {option.so\_luong\_ton}

</Typography>

<Typography variant="caption" color="text.secondary">

Giá bán: {option.gia\_ban.toLocaleString()} VND

</Typography>

</Box>

</li>

)}

noOptionsText="Không tìm thấy thuốc nào"

filterOptions={(options, { inputValue }) =>

options.filter((option) =>

option.ten\_thuoc

.toLowerCase()

.includes(inputValue.toLowerCase())

)

}

/>

{selectedMedicine && (

<>

<Paper

elevation={1}

sx={{

p: 2,

backgroundColor: "grey.50",

borderRadius: 1,

}}

>

<Typography

variant="subtitle2"

sx={{ fontWeight: 600, mb: 1 }}

>

Thuốc đã chọn: {selectedMedicine.ten\_thuoc}

</Typography>

<Typography variant="body2" color="text.secondary">

Đơn vị: {selectedMedicine.don\_vi}

</Typography>

<Typography variant="body2" color="text.secondary">

Số lượng hiện tại: {selectedMedicine.so\_luong\_ton}{" "}

{selectedMedicine.don\_vi}

</Typography>

<Typography variant="body2" color="text.secondary">

Đường dùng: {selectedMedicine.duong\_dung}

</Typography>

</Paper>

<Divider sx={{ my: 1 }} />

<Typography variant="subtitle2" sx={{ fontWeight: 600 }}>

Nhập hàng và cập nhật thông tin

</Typography>

<TextField

label="Số lượng nhập thêm"

type="number"

value={additionalQuantity}

onChange={handleAdditionalQuantityChange}

inputProps={{ min: 0 }}

helperText="Nhập 0 nếu chỉ muốn cập nhật thông tin"

fullWidth

/>

<Box sx={{ display: "flex", gap: 2 }}>

<TextField

label="Giá nhập (VND)"

type="number"

value={editingData.gia\_nhap}

onChange={handleEditingDataChange("gia\_nhap")}

inputProps={{ min: 0 }}

fullWidth

/>

<TextField

label="Giá bán (VND)"

type="number"

value={editingData.gia\_ban}

onChange={handleEditingDataChange("gia\_ban")}

inputProps={{ min: 0 }}

fullWidth

/>

</Box>

<TextField

label="Số lượng đặt hàng"

type="number"

value={editingData.so\_luong\_dat\_hang}

onChange={handleEditingDataChange("so\_luong\_dat\_hang")}

inputProps={{ min: 0 }}

fullWidth

/>

{(additionalQuantity > 0 || hasChanges) && (

<Paper

elevation={0}

sx={{

p: 2,

backgroundColor: "info.light",

color: "info.contrastText",

borderRadius: 1,

}}

>

<Typography variant="body2" sx={{ mb: 1, fontWeight: 500 }}>

Thay đổi sẽ được áp dụng:

</Typography>

{additionalQuantity > 0 && (

<Typography variant="body2">

• Số lượng sau khi nhập:{" "}

{selectedMedicine.so\_luong\_ton + additionalQuantity}{" "}

{selectedMedicine.don\_vi}

</Typography>

)}

{editingData.gia\_nhap !== selectedMedicine.gia\_nhap && (

<Typography variant="body2">

• Giá nhập: {selectedMedicine.gia\_nhap.toLocaleString()}{" "}

→ {editingData.gia\_nhap.toLocaleString()} VND

</Typography>

)}

{editingData.gia\_ban !== selectedMedicine.gia\_ban && (

<Typography variant="body2">

• Giá bán: {selectedMedicine.gia\_ban.toLocaleString()} →{" "}

{editingData.gia\_ban.toLocaleString()} VND

</Typography>

)}

{editingData.so\_luong\_dat\_hang !==

selectedMedicine.so\_luong\_dat\_hang && (

<Typography variant="body2">

• Số lượng đặt hàng:{" "}

{selectedMedicine.so\_luong\_dat\_hang} →{" "}

{editingData.so\_luong\_dat\_hang}

</Typography>

)}

</Paper>

)}

</>

)}

</Box>

)}

</DialogContent>

<DialogActions>

<Button onClick={handleClose} color="secondary">

Hủy

</Button>

<Button

onClick={handleSubmit}

variant="contained"

color="primary"

disabled={isSubmitDisabled()}

>

{activeTab === 0 ? "Lưu" : hasChanges ? "Cập nhật" : "Đóng"}

</Button>

</DialogActions>

</Dialog>

);

};

export default AddMedicineDialog;