// src/components/Dashboard.js

import React, { useState } from 'react';

import { Grid, Paper, Typography, TextField, Button, IconButton, Dialog, DialogTitle, DialogContent, DialogActions, Box } from '@mui/material';

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

import Notification from './Notification'; // Import the Notification component

import NotificationsIcon from '@mui/icons-material/Notifications';

import {

Chart as ChartJS,

CategoryScale,

LinearScale,

BarElement,

Title,

Tooltip,

Legend,

ArcElement,

} from 'chart.js';

// Register the components

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

const Dashboard = () => {

const [items, setItems] = useState([

{ name: 'Laptop', price: 999, category: 'Electronics', image: 'https://www.livemint.com/lm-img/img/2024/06/18/600x338/laptoppppp\_cleanup\_1718705991705\_1718706003867.PNG' },

{ name: 'Shirt', price: 29, category: 'Clothing', image: 'https://questanews.com/wp-content/uploads/2022/03/shop-clothing-clothes-shop-hanger-modern-shop-boutique-scaled-e1648594589180.jpg' },

{ name: 'Apple', price: 1, category: 'Groceries', image: 'https://thumbs.dreamstime.com/b/lots-groceries-17001094.jpg' },

]);

const [notifications, setNotifications] = useState([]);

const [newItem, setNewItem] = useState({ name: '', price: '', category: '' });

const [dialogOpen, setDialogOpen] = useState(false); // State for notification dialog

// Sample data for the charts

const getPieChartData = () => {

const categoryCounts = items.reduce((acc, item) => {

acc[item.category] = (acc[item.category] || 0) + 1;

return acc;

}, {});

return {

labels: Object.keys(categoryCounts),

datasets: [

{

data: Object.values(categoryCounts),

backgroundColor: ['#FF6384', '#36A2EB', '#FFCE56'],

},

],

};

};

const getBarChartData = () => {

const stockCounts = items.reduce((acc, item) => {

acc[item.name] = (acc[item.name] || 0) + 1; // Just for demo purposes, can adjust to real quantities

return acc;

}, {});

return {

labels: Object.keys(stockCounts),

datasets: [

{

label: 'Stock Quantity',

data: Object.values(stockCounts),

backgroundColor: '#42A5F5',

},

],

};

};

const handleAddItem = () => {

if (!newItem.name || !newItem.price || !newItem.category) {

setNotifications([...notifications, { message: 'Please fill all fields!', type: 'error' }]);

return;

}

const item = {

name: newItem.name,

price: parseFloat(newItem.price),

category: newItem.category,

image: 'https://via.placeholder.com/100', // Placeholder image

};

setItems([...items, item]);

setNotifications([...notifications, { message: ${item.name} added successfully!, type: 'success' }]);

setNewItem({ name: '', price: '', category: '' }); // Reset form

};

// Handle open dialog

const handleOpenDialog = () => {

setDialogOpen(true);

};

// Handle close dialog

const handleCloseDialog = () => {

setDialogOpen(false);

};

return (

<Grid container spacing={3} style={{ padding: '20px' }}>

<Grid item xs={12}>

<Box display="flex" alignItems="center" justifyContent="space-between">

<Typography variant="h4" gutterBottom>

Dashboard

</Typography>

<IconButton color="primary" onClick={handleOpenDialog}>

<NotificationsIcon />

</IconButton>

</Box>

</Grid>

<Grid item xs={12}>

<Paper elevation={3} style={{ padding: '20px' }}>

<Typography variant="h6" gutterBottom>

Add Item

</Typography>

<Grid container spacing={2}>

<Grid item xs={4}>

<TextField

label="Item Name"

variant="outlined"

fullWidth

value={newItem.name}

onChange={(e) => setNewItem({ ...newItem, name: e.target.value })}

/>

</Grid>

<Grid item xs={4}>

<TextField

label="Price"

variant="outlined"

fullWidth

value={newItem.price}

onChange={(e) => setNewItem({ ...newItem, price: e.target.value })}

/>

</Grid>

<Grid item xs={4}>

<TextField

label="Category"

variant="outlined"

fullWidth

value={newItem.category}

onChange={(e) => setNewItem({ ...newItem, category: e.target.value })}

/>

</Grid>

</Grid>

<Button

variant="contained"

color="primary"

onClick={handleAddItem}

style={{ marginTop: '20px' }}

>

Add Item

</Button>

</Paper>

</Grid>

<Grid item xs={12} md={6}>

<Paper elevation={3} style={{ padding: '20px' }}>

<Typography variant="h6" gutterBottom>

Item Categories Distribution

</Typography>

<div style={{ width: '300px', height: '300px', margin: '0 auto' }}>

<Pie data={getPieChartData()} options={{ responsive: true, maintainAspectRatio: false }} />

</div>

</Paper>

</Grid>

<Grid item xs={12} md={6}>

<Paper elevation={3} style={{ padding: '20px' }}>

<Typography variant="h6" gutterBottom>

Stock Quantity

</Typography>

<div style={{ width: '300px', height: '300px', margin: '0 auto' }}>

<Bar data={getBarChartData()} options={{ responsive: true, maintainAspectRatio: false }} />

</div>

</Paper>

</Grid>

<Grid item xs={12}>

<Paper elevation={3} style={{ padding: '20px' }}>

<Typography variant="h6" gutterBottom>

Stock Items

</Typography>

<ul style={{ listStyleType: 'none', padding: 0 }}>

{items.map((item, index) => (

<li key={index} style={{ display: 'flex', alignItems: 'center', marginBottom: '10px' }}>

<img src={item.image} alt={item.name} style={{ width: '50px', marginRight: '10px' }} />

<Typography variant="body1">

{item.name} - ${item.price}

</Typography>

</li>

))}

</ul>

</Paper>

</Grid>

<Grid item xs={12}>

<Paper elevation={3} style={{ padding: '20px' }}>

<Notification notifications={notifications} /> {/\* Pass notifications to Notification component \*/}

</Paper>

</Grid>

{/\* Notification Dialog \*/}

<Dialog open={dialogOpen} onClose={handleCloseDialog}>

<DialogTitle>Notifications</DialogTitle>

<DialogContent>

{notifications.length === 0 ? (

<Typography>No notifications</Typography>

) : (

notifications.map((notification, index) => (

<Typography key={index} style={{ color: notification.type === 'error' ? 'red' : 'green' }}>

{notification.message}

</Typography>

))

)}

</DialogContent>

<DialogActions>

<Button onClick={handleCloseDialog} color="primary">

Close

</Button>

</DialogActions>

</Dialog>

</Grid>

);

};

export default Dashboard;