//Vehicles.jsx infinity scroll

import Header from "../components/Header";

import VehicleCard from "../components/VehicleCard";

import Filter from "../components/Filter";

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

import Loader from '../components/Loader.jsx';

import API\_URL from "../API\_URL.js";

export default function Vehicles() {

  const [vehicles, setVehicles] = useState([]);

  const [isLoading, setIsLoading] = useState(false);

  const [filteredVehicles, setFilteredVehicles] = useState([]);

  const [error, setError] = useState(null);

  const [filterBrands, setFilterBrands] = useState([]);

  const [filterCarType, setFilterCarType] = useState([]);

  const [page, setPage] = useState(1);

  const [hasMore, setHasMore] = useState(true);

  const loadVehicles = useCallback(async () => {

    if (isLoading || !hasMore) return;

    setIsLoading(true);

    try {

      const response = await fetch(`${API\_URL}/Vehicles?page=${page}&limit=`);

      if (!response.ok) {

        setError("Error: Došlo je do greške prilikom preuzimanja vozila");

        return;

      }

      const resData = await response.json();

      if (!Array.isArray(resData.vehicles)) {

        setError("Error: Došlo je do greške sa serverom, očekivani podaci nisu ispravni.");

        return;

      }

      if (resData.vehicles.length === 0) {

        setHasMore(false);

      }

      setVehicles((prevVehicles) => [...prevVehicles, ...resData.vehicles]);

      setFilteredVehicles((prevVehicles) => [

        ...prevVehicles,

        ...resData.vehicles.filter((vehicle) => vehicle.status === "Dostupno"),

      ]);

    } catch (e) {

      setError("Error: Došlo je do greške od strane servera.");

    } finally {

      setIsLoading(false);

    }

  }, [page]);

  useEffect(() => {

    loadVehicles();

  }, [page, loadVehicles]);

  const applyFilters = (filters) => {

    const newFilteredVehicles = vehicles.filter((vehicle) => {

      const selectedPrice = filters.price ? filters.price.split(',').map(Number) : null;

      return (

        vehicle.status === "Dostupno" &&

        (filters.brand ? vehicle.brand === filters.brand : true) &&

        (filters.type ? vehicle.type === filters.type : true) &&

        (filters.transmission ? vehicle.transmission === filters.transmission : true) &&

        (selectedPrice

          ? vehicle.pricePerDay >= selectedPrice[0] && vehicle.pricePerDay <= selectedPrice[1]

          : true) &&

        (filters.fuel ? vehicle.fuelType === filters.fuel : true) &&

        (filters.doors ? vehicle.numOfDoors === parseInt(filters.doors) : true)

      );

    });

    setFilteredVehicles(newFilteredVehicles);

  };

  const handleScroll = () => {

    if (isLoading || !hasMore) return;

    const bottom = window.innerHeight + document.documentElement.scrollTop === document.documentElement.offsetHeight;

    if (bottom) {

      setPage((prevPage) => prevPage + 1);

    }

  };

  useEffect(() => {

    window.addEventListener("scroll", handleScroll);

    return () => {

      window.removeEventListener("scroll", handleScroll);

    };

  }, [isLoading, hasMore]);

  return (

    <>

      <Header title="Odaberite vozilo" />

      <Filter

        filterBrands={filterBrands}

        filterCarType={filterCarType}

        onFilterChange={applyFilters}

      />

      {isLoading && <Loader />}

      {error && <p className="error-message">{error}</p>}

      {!isLoading && !error && filteredVehicles.length === 0 && (

        <p className="center">Nemamo vozilo po tim zahtevima!</p>

      )}

      {!isLoading && !error && filteredVehicles.length > 0 && (

        <div className="vehicles">

          {filteredVehicles.map((vehicle) => (

            <VehicleCard key={vehicle.id + page} vehicle={vehicle} />

          ))}

        </div>

      )}

    </>

  );

}