// MenuComponent.tsx

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

import {getMeny, languages, currentLanguage, toggleLanguage } from "../../data/Meny/LanguageDecider";

import {DishProps, MenuCategory} from "../../types"

const DishComponent: React.FC<DishProps> = ({ dish }) => {

  const [currentImageIndex, setCurrentImageIndex] = useState(0);

  const [selectedPiece, setSelectedPiece] = useState<number | null>(dish.pieces?.[0] || null);

  const [selectedSize, setSelectedSize] = useState<string | null>(dish.size?.[0] || null);

  const [selectedType, setSelectedType] = useState<string | null>(dish.types?.[0] || null);

  const [selectedPrice, setSelectedPrice] = useState<number | null>(

    dish.prices.length > 0 ? dish.prices[0] : null

  );

  const [selectedDishName, setSelectedDishName] = useState<string | null>(

    dish.dishname.length > 0 ? dish.dishname[0] : null

  );

  const [selectedDescription, setSelectedDescription] = useState<string | null>(

    (dish.description || []).length > 0 ? dish.description![0] : null

  );

  const getDescription = (index: number) => {

    if (dish.description && dish.description.length > 0) {

      if (dish.description.length === 1) {

        return dish.description[0];

      } else {

        return dish.description[index] || null;

      }

    }

    return null;

  };

  const handlePiecesClick = (piece: number, index: number) => {

    if (dish.images) {

      setCurrentImageIndex(index);

      setSelectedPiece(piece);

      setSelectedSize(null);

      setSelectedType(null);

      setSelectedPrice(dish.prices.length === 1 ? dish.prices[0] : dish.prices[index]);

      setSelectedDishName(dish.dishname.length === 1 ? dish.dishname[0] : dish.dishname[index]);

      setSelectedDescription(getDescription(index));

    }

  };

  const handleSizeClick = (size: string, index: number) => {

    if (dish.images) {

      setCurrentImageIndex(index);

      setSelectedSize(size);

      setSelectedPiece(null);

      setSelectedType(null);

      setSelectedPrice(dish.prices.length === 1 ? dish.prices[0] : dish.prices[index]);

      setSelectedDishName(dish.dishname.length === 1 ? dish.dishname[0] : dish.dishname[index]);

      setSelectedDescription(getDescription(index));

    }

  };

  const handleTypeClick = (type: string, index: number) => {

    if (dish.images) {

      setCurrentImageIndex(index);

      setSelectedType(type);

      setSelectedPiece(null);

      setSelectedSize(null);

      setSelectedPrice(dish.prices.length === 1 ? dish.prices[0] : dish.prices[index]);

      setSelectedDishName(dish.dishname.length === 1 ? dish.dishname[0] : dish.dishname[index]);

      setSelectedDescription(getDescription(index));

    }

  };

  const uniquePrices = [...new Set(dish.prices)];

  const [isModalOpen, setIsModalOpen] = useState(false);

  const handleImageClick = () => {

    setIsModalOpen(!isModalOpen);

  };

  const handleModalClose = () => {

    setIsModalOpen(false);

  };

  return (

    <div key={dish.dishname[0]} className="rounded-xl flex flex-col justify-between h-full border-2 border-red-900">

      <section className="">

        <div className="flex flex-col">

          <div className="bg-red-900 rounded-t-lg">

            <h3 className="text-2xl text-center font-bold font-cormorant text-white">{selectedDishName}</h3>

          </div>

          {dish.images && (

            <img

              className="max-h-fit cursor-pointer"

              onClick={handleImageClick}

              src={dish.images[currentImageIndex].toString()}

              alt={selectedDishName || ""}

            />

          )}

        </div>

        <p className="text-white h-16 text-center font-opensans my-2">{selectedDescription}</p>

      </section>

      {isModalOpen && dish.images && dish.images.length > 0 && (

  <dialog open className="fixed inset-0 min-h-full min-w-fit bg-black bg-opacity-90" onClick={handleModalClose}>

      <section className="mt-16 flex flex-col items-center">

          <div className="mt-24 lg:mt-24 xl:mt-12 flex flex-col">

      <img

        className="cursor-pointer lg:mx-52"

        onClick={handleImageClick}

        src={dish.images[currentImageIndex]?.toString()}

        alt={dish.dishname[0]}

              />

              </div>

      <div className="bg-red-900 max-w-fit rounded-lg p-1 text-white font-cormorant font-bold text-center">

        <p className="text-md md:text-lg lg:text-xl xl:text-2xl 2xl:text-3xl">

          {selectedDishName}

        </p>

        {selectedType || selectedPiece || selectedSize ? (

    <p className="text-xs md:text-md lg:text-lg xl:text-xl 2xl:text-2xl">

      {selectedType || (selectedPiece ? selectedPiece + " Bitar" : null) || selectedSize}

    </p>

  ) : null}

              </div>

              </section>

  </dialog>

)}

      <section className="font-opensans text-white">

        {dish.types && dish.types.length > 0 && (

    <div

    className={`${

      dish.types.length < 4 ? 'flex justify-center items-center' : 'grid'

    } grid-cols-4 gap-0`}

  >

    {dish.types.map((types, index) => (

        <span

          key={types}

          onClick={() => handleTypeClick(types, index)}

          className={`${

            selectedType === types ? "bg-red-900" : "transparent"

          } mx-1 p-1 rounded-lg text-center cursor-pointer`}

        >

          {types}{" "}

        </span>

      ))}

    </div>

  )}

    {dish.pieces && dish.pieces.length > 0 && (

    <div

    className={`${

      dish.pieces.length < 4 ? 'flex justify-center items-center' : 'grid'

    } grid-cols-4 gap-0`}

  >

            {dish.pieces.map((piece, index) => (

        <span

          key={piece}

          onClick={() => handlePiecesClick(piece, index)}

          className={`${

            selectedPiece === piece ? "bg-red-900" : "transparent"

          } mx-1 p-1 rounded-lg text-center cursor-pointer`}

        >

          {piece}{" "} Bitar

        </span>

      ))}

      </div>

  )}

    {dish.size && dish.size.length > 0 && (

    <div

    className={`${

      dish.size.length < 4 ? 'flex justify-center items-center' : 'grid'

    } grid-cols-4 gap-0`}

  >

      {dish.size.map((size, index) => (

        <span

          key={size}

          onClick={() => handleSizeClick(size, index)}

          className={`${

            selectedSize === size ? "bg-red-900" : "transparent"

          } mx-1 p-1 rounded-lg text-center cursor-pointer`}

        >

          {size}{" "}

        </span>

      ))}

    </div>

          )}

<div className={`${

  uniquePrices.length < 4 ? "flex flex-row flex-wrap gap-x-6 mx-2 my-4 justify-center" : "grid grid-cols-4 gap-0  justify-center m-2"

} `}>

          {uniquePrices.map((price) => (

            <span

              key={price}

              className={`${selectedPrice === price ? "bg-red-900" : "transparent"} mx-1 p-1 text-center text-lg md:text-base rounded-lg`}

            >

              {price}kr

            </span>

          ))}

        </div>

      </section>

    </div>

  );

};

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

  useEffect(() => {

    // Do something when the language changes

    console.log("Language changed in MenuComponent:", currentLanguage);

  }, [currentLanguage]);

  const selectedMeny = getMeny() as MenuCategory[];

  /\* const Meny = languages[currentLanguage];

  console.log("Current language in Meny:", currentLanguage);

  console.log("Meny:", Meny); \*/

  return (

    <div className="container mx-auto my-32">

      {selectedMeny.map((category) => (

        <div key={category.name} className="my-40">

          <section className="flex flex-col mx-8 my-8 lg:my-0">

            <h2 className="text-6xl font-cormorant text-golden text-center font-bold my-8" id={category.id}>

              {category.name}

            </h2>

            <div

              className={

                category.dishes.length < 4

                  ? "flex flex-col lg:flex-row justify-center gap-8 max-w-fit"

                  : "grid grid-cols-1 md:grid-cols-2 2xl:grid-cols-4 col-end-auto gap-8"

              }

            >

              {category.dishes.map((dish) => (

                <div key={dish.id} className="container">

                  <DishComponent dish={dish} />

                </div>

              ))}

            </div>

          </section>

        </div>

      ))}

    </div>

  );

};

export default MenuComponent;

// LanguageSwitcher.tsx

import React from 'react';

import flag from "../../assets/flags/SWE-UK.webp"

import { toggleLanguage } from '../../data/Meny/LanguageDecider';

const LanguageSwitcher: React.FC<{ onLanguageChange: () => void }> = ({ onLanguageChange }) => {

  const handleLanguageChange = () => {

    toggleLanguage();

    onLanguageChange();

  };

  return (

    <div className="w-20">

      <img

        src={flag}

        alt="Swedish & UK Flag"

        className="cursor-pointer"

        onClick={handleLanguageChange}

      />

    </div>

  );

};

export default LanguageSwitcher;

// LanguageDecider.tsx

import { useState } from "react";

// SWEDISH MENY

import { sweLunch } from "./Swedish/Lunch";

import { sweSushi } from "./Swedish/Sushi";

import { sweNorimaki } from "./Swedish/Norimaki";

import { sweSashimi } from "./Swedish/Sashimi";

import { swePokebowl } from "./Swedish/Pokebowl";

import { sweVarmratter } from "./Swedish/Varmratter";

import { sweSpecial } from "./Swedish/Special";

import { sweBarn } from "./Swedish/Barn";

//ENGLISH MENY

import { engLunch } from "./English/Lunch";

import { engSushi } from "./English/Sushi";

import { engNorimaki } from "./English/Norimaki";

import { engSashimi } from "./English/Sashimi";

import { engPokebowl } from "./English/Pokebowl";

import { engVarmratter } from "./English/Varmratter";

import { engSpecial } from "./English/Special";

import { engBarn } from "./English/Barn";

const languages = {

    swedish: [sweLunch, sweSushi, sweNorimaki, sweSashimi, swePokebowl, sweVarmratter, sweSpecial, sweBarn],

    english: [engLunch, engSushi,engNorimaki, engSashimi, engPokebowl, engVarmratter, engSpecial, engBarn],

    french: [],

    spanish: [],

    japanese: [],

  };

  const defaultLanguage = "swedish";

  interface LanguageDeciderProps {

    getMeny: () => typeof languages[keyof typeof languages];

    currentLanguage: string;

    toggleLanguage: () => void;

  }

  const LanguageDecider: React.FC<LanguageDeciderProps> = () => {

    const [currentLanguage, setCurrentLanguage] = useState(defaultLanguage);

    const toggleLanguage = () => {

      const newLanguage = currentLanguage === "swedish" ? "english" : "swedish";

      console.log(`Switching to ${newLanguage}`);

      setCurrentLanguage(newLanguage);

    };

    const getMeny = () => languages[currentLanguage];

    return {

      getMeny,

      currentLanguage,

      toggleLanguage,

    };

  };

  const { getMeny, currentLanguage, toggleLanguage } = LanguageDecider();

  export { languages, getMeny, currentLanguage, toggleLanguage };

/\* import {Lunch} from "./Lunch";

import {Sushi} from "./Sushi";

import {Norimaki} from "./Norimaki";

import {Sashimi} from "./Sashimi";

import {Pokebowl} from "./Pokebowl";

import {Varmratter} from "./Varmratter";

import {Special} from "./Special";

import { Barn } from "./Barn"; \*/

/\* import { MenuCategory } from "../../types";

const swedishMeny: MenuCategory[] = [sweLunch, sweSushi, sweNorimaki, sweSashimi, swePokebowl, sweVarmratter, sweSpecial, sweBarn];

const englishMeny: MenuCategory[] = [engLunch, engSushi, engNorimaki, engSashimi, engPokebowl, engVarmratter, engSpecial, engBarn]; \*/