Cart.tsx

import React from "react";

import { useContext } from "react";

import { ProductContext } from "../../../App";

import { AnimatePresence, motion } from "framer-motion";

import useCartLogic from "../../../hooks/useCartLogic";

import Alert from "../../data/Alert";

import { CartTotal } from "./CartTotal";

interface cartType {

  showCart: boolean;

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

}

enum ModeValue {

  INCREMENT = "INCREMENT",

  DECREMENT = "DECREMENT",

}

const Cart = (props: cartType) => {

  const {

    itemsSelected,

    setItemsSelected,

    selectedItemQuantity,

    setSelectedItemQuantity,

*//  setSelectedItemsCount,*

  } = useContext(ProductContext);

  const {

    showAlert,

    setShowAlert,

    removeFromCart,

    changeQuantity,

  } = useCartLogic(itemsSelected, setItemsSelected, setSelectedItemQuantity);

  return (

    <>

      <Alert

        msg="REMOVED FROM CART"

        showAlert={showAlert}

        setShowAlert={setShowAlert}

      />

      <AnimatePresence>

        {props.showCart && (

          <>

            <motion.button

              initial={{ y: "-100vh" }}

              animate={{ y: 0 }}

              exit={{ y: "-100vh" }}

              transition={{ type: "tween", delay: 0.1 }}

              onClick={() => props.setShowCart((showCart) => false)}

              className="fixed right-10 top-4 z-[70] text-3xl  text-slate-700 pb-3 text-center bg-slate-100 hover:bg-slate-300 w-10 h-10 rounded-full "

            >

              &times;

            </motion.button>

            <motion.div

              initial={{ x: "100vw" }}

              animate={{ x: 0 }}

              exit={{ x: "100vw" }}

              transition={{ type: "spring", damping: 20 }}

              className="fixed  overflow-y-scroll z-[60] w-4/5 md:w-2/3 lg:w-1/2 right-0 top-0 text-white

            bg-slate-200 flex flex-col items-center shadow-lg shadow-slate-600  h-full "

            >

              <h1 className="text-5xl font-bold text-slate-800  h-[12%] text-center pt-3">

                Your Items

              </h1>

              <AnimatePresence>

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

                  <motion.div

                    initial={{ x: "-100vw" }}

                    animate={{ x: 0 }}

                    exit={{ x: "-100vw" }}

                    transition={{ type: "tween", delay: 0.1 \* index }}

                    className=" w-10/12 m-2 my-10 flex  h-2/5 shadow-lg shadow-slate-400 text-slate-800"

                    key={index}

                  >

                    <img

                      className=" h-full w-1/2 object-contain"

                      src={item.img\_url}

                      alt="image"

                    />

                    <div className="p-5 pt-1 text-lg font-semibold w-full">

                      <h1 className="text-xl  font-bold">{item.type}</h1>

                      <h2 className="text-xl font-semibold">{item.category}</h2>

                      <p className="mt-2">Size: {item.size}</p>

                      <div className="flex my-1 ">

                        Color:

                        <p

                          className={`mx-2 mt-[2px]  w-6 h-6 border border-slate-500 rounded-full`}

                          style={{ backgroundColor: item.color }}

                        />

                      </div>

                      <div className="mt-4 flex justify-between items-center">

                        <p className="text-lg font-bold">

                          $

                          {Number(

                            item.price \* selectedItemQuantity[index]

                          ).toFixed(2)}

                        </p>

                        <div>

                          <button

*//put it inside a function*

                            onClick={() =>

                              changeQuantity(

                                ModeValue.DECREMENT,

                                selectedItemQuantity,

                                item,

                                index

                              )

                            }

                            className="text-2xl font-bold"

                          >

                            &#8722;

                          </button>

                          <input

                            value={selectedItemQuantity[index]}

                            min={1}

                            type="number"

                            id="myNumberInput"

                            readOnly

                            className="w-14 py-1 mx-2 outline-none text-center   bg-slate-100"

                          />

                          <button

                            onClick={(e) =>

                              changeQuantity(

                                ModeValue.INCREMENT,

                                selectedItemQuantity,

                                item,

                                index

                              )

                            }

                            className="text-2xl font-bold"

                          >

                            &#43;

                          </button>

                        </div>

                      </div>

                      <button

                        onClick={() => {

*// setSelectedItemsCount((prev) => prev - 1);*

*// setShowAlert(showAlert=>true)*

                          removeFromCart(index, item, selectedItemQuantity);

                        }}

                        className=" font-bold bg-slate-900 text-xs mt-4 w-full text-slate-300 rounded-lg p-2"

                      >

                        REMOVE FROM CART

                      </button>

                    </div>

                  </motion.div>

                ))}

              </AnimatePresence>

              <CartTotal />

            </motion.div>

          </>

        )}

      </AnimatePresence>

    </>

  );

};

export default Cart;

footer.tsx

import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";

import { faEnvelope, faPhone, faAddressBook } from "@fortawesome/free-solid-svg-icons";

import { library } from "@fortawesome/fontawesome-svg-core";

library.add(faEnvelope,faPhone,faAddressBook)

import { motion } from "framer-motion";

import React from "react";

import Logo from "../../utils/Logo";

const Footer = () => {

  const getYear = () => {

    const date = new Date();

    return date.getFullYear();

  };

  const footerPStyle1= "ml-14 hover:color:slate-300 my-2 origin-left transition-all duration-300 hover:scale-105"

  const footerPStyle2= "ml-7 hover:color:slate-300 my-2 origin-left transition-all duration-300 hover:scale-105"

  return (

    <>

      <div className="relative h-full items-center flex flex-col lg:items-baseline lg:flex-row lg:h-vh-80 w-full bg-black">

        <section className="w-1/2 h-5/6 flex flex-col items-center">

          <div className="flex mt-10 mr-20">

            <Logo

              height={80}

              width={80}

              mr={8}

              mt={1}

              ml={0}

              mb={0}

              textMarginTop={0}

              logoColor="white"

              textSize={"5xl"}

            />

          </div>

          <p className="text-white ml-10 text-sm">

            ULTIMATE SHOPPING DESTINATION

          </p>

          <div className="mt-6 h-full w-2/3 text-center text-white">

            {/\* add email \*/}

            <p>

              Contact us with any questions or feedback and our customer care

              team will be happy to assist you

            </p>

            <p className="mt-8 text-lg ">

              <FontAwesomeIcon icon="envelope" className="mx-2" />

              vibekart@gmail.com

            </p>

            <p className="my-2 text-lg">

              <FontAwesomeIcon icon="phone" className="mx-2" />

              +92-123-4567891

            </p>

            <p className="my-2 text-lg">

              <FontAwesomeIcon icon="address-book" className="mx-2" />

              456 Maple Avenue, Springfield, USA 67890

            </p>

          </div>

          <button

            className="text-white absolute bottom-1 md:bottom-2 lg:bottom-32 xl:bottom-20 opacity-70 text-lg "

            onClick={() => {

              window.scrollTo({

                top: 0,

                behavior: "smooth",

              });

            }}

          >

            Scroll to top

          </button>

        </section>

        <section className="mt-7 lg:flex flex-col items-center h-5/6 w-full lg:w-1/2 text-white p-12  text-4xl font-bold">

          <div className=" h-full  flex md:flex-row ml-10 items-center flex-col">

            <div className="w-full h-full mx-2   mb-10">

              <h1 className=" border-l-8 ml-10 border-slate-400 pl-3">

                VibeKart

              </h1>

              <div className="text-sm lg:text-xl font-normal mt-4 ">

                <p className={`${footerPStyle1}`}>Who are we</p>

                <p className={`${footerPStyle1}`}>Contact</p>

                <p className={`${footerPStyle1}`}>Our impact</p>

                <p className={`${footerPStyle1}`}>Terms & conditions</p>

                <p className={`${footerPStyle1}`}>Privacy policy</p>

                <p className={`${footerPStyle1}`}>How to shop</p>

              </div>

            </div>

            <div className="w-full h-full  mt-1">

              <h1 className="border-l-8 ml-3 border-slate-400 pl-3">

                Work with us

              </h1>

              <div className="text-sm lg:text-xl font-normal mt-5">

                <p className={`${footerPStyle2}`}>Sell with us</p>

                <p className={`${footerPStyle2}`}>How to get started</p>

                <p className={`${footerPStyle2}`}>Partner testimonal</p>

                <p className={`${footerPStyle2}`}>Our Partner support</p>

              </div>

            </div>

          </div>

        </section>

        {/\* <footer className=" absolute bottom-5  text-xl xl:text-3xl lg:left-1/3 text-white">

          <p>Copyright &#169; {getYear()}. All rights reserved.</p>

        </footer> \*/}

      </div>

    </>

  );

};

export default Footer;

MainPage.tsx

import { motion } from "framer-motion";

import { Suspense } from "react";

import { useNavigate } from "react-router-dom";

import { useTypewriter, Cursor } from "react-simple-typewriter";

import useGetImage from "../../hooks/useGetImage";

import Logo from "../../utils/Logo";

import Footer from "./Footer";

const Header = () => {

  const { img } = useGetImage();

  const navigate = useNavigate();

  const [text] = useTypewriter({

    words: [

      "Shop the world at your fingertips!",

      "Stay up to date with the latest trends and styles!",

      "At VibeKart, customer satisfaction is our top priority.",

      "We believe, customer loyalty is earned through exceptional service.",

      "We take pride in providing top-notch customer service to our valued shoppers.",

    ],

    loop: true,

    typeSpeed: 60,

    deleteSpeed: 40,

  });

  return (

    <>

      <main className="h-screen grid grid-cols-2 ">

        <div className="z-50  w-4/5 flex items-center font-bold justify-center text-lg lg:text-5xl text-white  text-shadow-100 ">

          <p

            aria-label="Loading messsages"

            role="status"

            className=" mb-16 lg:mb-20 lg:ml-24 justify-center text-center p-3 "

          >

            {text}

            <Cursor />

          </p>

        </div>

        <motion.img

          initial={{ x: "-100vw" }}

          animate={{ x: 0 }}

          transition={{ delay: 0.4, ease: "easeInOut" }}

          src={img}

          loading="lazy"

          className=" object-cover absolute top-0 h-full shadow-inner shadow-black w-full opacity-90 overflow-hidden"

        />

        <div className="bg-black absolute opacity-40 h-full w-full  text-6xl" />

        <div className="relative ">

          <div className="absolute md:top-20 top-10 right-16  md:left-20 p-2 ">

            <Logo

              width={90}

              height={90}

              ml={20}

              mr={8}

              mt={0}

              mb={0}

              textSize={'5xl'}

              logoColor="white"

            />

          </div>

          <button

            onClick={() => navigate("/mall")}

            className="mr-20 absolute top-[40%] md:left-24 border text-white text-4xl md:text-6xl border-white rounded-md py-4 px-10 md:p-6"

          >

            Shop Now

          </button>

        </div>

      </main>

      <Suspense fallback="Footer...">

        <Footer />

      </Suspense>

    </>

  );

};

export default Header;

ProductDisplay.tsx

interface productType {

  category: string;

  cloth\_type?: string;

  shoe\_type?: string;

  color: string[];

  createdAt: Date;

  description: string;

  img\_url: string;

  price: Record<string, string>;

  quantity: number;

  size: number[];

  updatedAt: Date;

  \_\_v: number;

  \_id: string;

}

export const ProductsDisplay = ({

  setClicked,

  setProductInfo,

  product,

}: {

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

  setProductInfo: React.Dispatch<React.SetStateAction<productType | undefined>>;

  product: productType;

}) => {

  return (

    product && (

      <div

        onClick={() => {

          setProductInfo(product);

          setClicked((clicked) => true);

        }}

        className=" hover:scale-105 bg-white transition-all duration-300 shadow-slate-400 shadow-xl flex flex-col items-center p-2 mx-2 my-10   h-[28rem] lg:w-80"

      >

        <img

          loading="lazy"

          src={product.img\_url}

          className=" mb-4 shadow-lg w-full h-[70%] object-contain rounded-lg"

        />

        <div className="w-full p-2">

          <p className="text-slate-500 text-xl font-bold tracking-wider">

            {product.cloth\_type || product.shoe\_type}

          </p>

          <p className="text-slate-400 font-bold  text-sm tracking-wide">

            {product.category}

          </p>

          <p className="text-slate-400 text-lg">

            $ {product.price.$numberDecimal}

          </p>

        </div>

      </div>

    )

  );

};

User.ts (backend)

import express from 'express'

import { UserModel } from "../../models/auth/User";

import dotenv from "dotenv";

dotenv.config();

import { StatusCodes } from "http-status-codes";

import { BadRequest, InternalServerError } from "../../errors";

export const register = async (req:express.Request, res: express.Response) => {

  const { email, password } = req.body

  if (!email || !password) {

    throw new BadRequest('Please provide the required credential')

  }

  const user = await UserModel.create({ ...req.body });

  const token = user.createJWT();

  res.status(StatusCodes.CREATED).send({ user: user.name, token });

};

export const login = async (req: express.Request, res:express.Response) => {

  const { username, email, password } = req.body;

  if ((!username && !email) || !password) {

    throw new BadRequest("Please provide required credentials");

  }

  //check via regex if user has entered username or an email on the frontend

  const user = username

    ? await UserModel.findOne({ username })

    : await UserModel.findOne({ email });

  if (!user) {

    throw new BadRequest("Invalid credentials");

  }

  const isValidPassword = await user.validatePassword(password);

  if (!isValidPassword) {

    throw new BadRequest("Invalid credentials");

  }

  const token = user.createJWT();

  res.status(StatusCodes.OK).json({ name: user.username, token });

};

Product.ts

import express from "express";

import { ClothModel } from "../../models/store/Cloth";

import { ShoeModel } from "../../models/store/Shoe";

import { FurnitureModel } from "../../models/store/Furniture";

import { CosmeticModel } from "../../models/store/Cosmetic";

import { ApplianceModel } from "../../models/store/Appliance";

import { DecorationModel } from "../../models/store/Decoration";

import "express-async-errors";

import { SortOrder } from "mongoose";

import { BadRequest } from "../../errors";

import { validate } from "./productValidation";

import QueryString, { ParsedQs } from "qs";

export const getAllProducts = async (

  req: express.Request,

  res: express.Response

) => {

  const { size }: { size?: string } = req.query;

  const { color }: { color?: string } = req.query;

  if (size) {

    const sz = size.split(",");

    req.query.size = sz;

  }

  if (color) {

    const clr = color.split(",");

    req.query.color = clr;

  }

  const DEFAULT\_SORT\_FIELD: string = "createdAt";

  const DEFAULT\_SORT\_VALUE: SortOrder = 1;

  const { type, sortOrder, page }: ParsedQs = req.query;

  validate(req.query);

  const sortBy = sortOrder

    ? { price: sortOrder }

    : { [DEFAULT\_SORT\_FIELD]: DEFAULT\_SORT\_VALUE };

  const validFields: Record<string, string> = {

    color: "$in",

    size: "$in", *// size can be equal to any value inside the array*

    category: "$eq",

  };

  const query: Record<string, Record<string, unknown>> = {};

  let result;

  let DATA\_DETAILS: Record<string,boolean> = {

    hasNext:false,

  }; *//to check if next page exist*

*//inserting queried parameters inside query object (makes code concise)*

  for (const [field, value] of Object.entries(req.query)) {

    if (validFields[field]) {

      query[field] = { [validFields[field]]: value };

    }

  }

  if (type) {

    result = await getModelData(type, sortBy, query, page, false, DATA\_DETAILS);

  } else {

*// const [clothes, shoes, furniture, appliances, decorations, cosmetics] =*

    const [clothes, shoes] = await Promise.all([

      getModelData("clothes", sortBy, {}, page, true, DATA\_DETAILS),

      getModelData("shoes", sortBy, {}, page, true, DATA\_DETAILS),

*// getModelData("furniture", sortBy, {}, page,true),*

*// getModelData("appliances", sortBy, {}, page,true),*

*// getModelData("decorations", sortBy, {}, page,true),*

*// getModelData("cosmetics", sortBy, {}, page,true),*

    ]);

*//only sending back cloth and shoe data as only it contains valid images*

    result = [

      ...clothes,

      ...shoes,

*// ...furniture,*

*// ...appliances,*

*// ...decorations,*

*// ...cosmetics,*

    ];

  }

*// DATA\_DETAILS.hasNext =result.length < DATA\_DETAILS.length*

  res.send({

    success: true,

    nbHits: result.length,

    hasNext: DATA\_DETAILS.hasNext,

    data: result,

  });

};

const getModelData = async (

  type: string | string[] | QueryString.ParsedQs | QueryString.ParsedQs[],

  sortBy: any,

  query: Record<string, unknown>,

  page:

    | string

    | QueryString.ParsedQs

    | string[]

    | QueryString.ParsedQs[]

    | undefined,

  sendAllDocuments: boolean,

  DATA\_DETAILS: Record<string,boolean>

) => {

  let result;

  let resultNext;

  const pageNumber = Number(page) || 1;

  const limit = sendAllDocuments ? 6 : 12;

  const skip = (pageNumber - 1) \* limit;

  const skipNext = pageNumber \* limit;

  switch (type) {

    case "clothes":

      result = ClothModel.find(query);

      resultNext = ClothModel.find(query);

      break;

    case "shoes":

      result = ShoeModel.find(query);

      resultNext = ShoeModel.find(query);

      break;

    case "appliances":

      result = ApplianceModel.find(query);

      resultNext = ApplianceModel.find(query);

      break;

    case "decorations":

      result = DecorationModel.find(query);

      resultNext = DecorationModel.find(query);

      break;

    case "cosmetics":

      result = CosmeticModel.find(query);

      resultNext = CosmeticModel.find(query);

      break;

    case "furniture":

      result = FurnitureModel.find(query);

      resultNext = FurnitureModel.find(query);

      break;

    default:

      throw new BadRequest(`Type, ${type} doesn't exist`);

  }

  const res = await resultNext.skip(skipNext).limit(limit).sort(sortBy);

  DATA\_DETAILS.hasNext = res.length > 0

  return await result.skip(skip).limit(limit).sort(sortBy);

};

ProductValidation.ts

import { ParsedQs } from "qs";

import { BadRequest } from "../../errors";

const clothCategory = [

  "Men shirts",

  "Men t-shirts",

  "Men trousers",

  "Men jeans",

  "Women t-shirts",

  "Women trousers",

  "Women jeans",

  "Women dresses",

];

const shoeCategory = [

  "Sneakers",

  "Dress Shoes",

  "Loafers",

  "Sandals",

  "Flats",

  "Sport shoes",

  "Heels",

];

const furnitureCategory = [

  "Sofas",

  "Chairs",

  "Sidetables",

  "TV stands",

  "Dining tables",

  "Dining chairs",

  "Buffets",

  "Bar stools",

  "Beds",

  "Headboards",

  "Vanities",

  "Dressers",

  "Desks",

  "Filing cabinets",

  "Bookcases",

];

const decorationCategory = [

  "Clocks",

  "Mirrors",

  "Paintings",

  "Table Vases",

  "Table Lamps",

  "Candles",

  "Area Rugs",

  "Floor Vases",

  "Floor Lamps",

  "Wreaths",

  "Christmas Lights",

  "Ornaments",

  "Spring Flowers",

  "Fall Leaves",

];

const cosmeticsCategory = [

  "Foundation",

  "Blush",

  "Highlighter",

  "Concealer",

  "Mascara",

  "Eyeliner",

  "Eyeshadow",

  "Lipstick",

  "Lip gloss",

  "Cleanser",

  "Moisturizer",

  "Serum",

  "Toner",

];

const applianceCategory = [

  "Refrigerators",

  "Freezers",

  "Ovens",

  "Microwaves",

  "Dishwashers",

  "Washing machines",

  "Combination washer/dryers",

  "Vacuum cleaners",

  "Steam cleaners",

  "Carpet cleaners",

];

export const validate = (queryParams: ParsedQs) => {

  const { type, category, color, size } = queryParams;

  if (!type && (category || color || size)) {

    throw new BadRequest("Type shall be identified");

  }

  switch (type) {

    case "furniture":

    case "cosmetics":

    case "appliances":

    case "decorations":

      if (color || size) {

        throw new BadRequest(`Invalid paramter on type ${type}`);

      }

      break;

    default:

      break;

  }

  if (category && !isValidCategory(type, category)) {

    throw new BadRequest(`Invalid category on type ${type}`);

  }

};

export const isValidCategory = (

  type: string | string[] | ParsedQs | ParsedQs[] | undefined,

  category: string | string[] | ParsedQs | ParsedQs[]

) => {

  switch (type) {

    case "clothing":

      return clothCategory.includes(category.toString());

    case "furniture":

      return furnitureCategory.includes(category.toString());

    case "cosmetics":

      return cosmeticsCategory.includes(category.toString());

    case "decoration":

      return decorationCategory.includes(category.toString());

    case "appliances":

      return applianceCategory.includes(category.toString());

    case "shoes":

      return shoeCategory.includes(category.toString());

  }

};