const increaseQuantity = (id, quantity, stock) => {

    const newQty = quantity + 1;

    if (stock <= quantity) {

      return;

    } else {

      dispatch(addItemToCart(id, newQty));

    }

  };

exports.newOrder = asyncWrapper(async (req, res, next) => {

  const {

    shippingInfo,

    orderItems,

    paymentInfo,

    itemsPrice,

    taxPrice,

    shippingPrice,

    totalPrice,

  } = req.body;

  const order = await orderModel.create({

    shippingInfo,

    orderItems,

    paymentInfo,

    itemsPrice,

    taxPrice,

    shippingPrice,

    totalPrice,

    user: req.user.\_id, // from authenticated user

    paidtAt: Date.now(),

  });

  res.status(201).json({

    success: true,

    order,

  });

});

router.route("/order/new").post(isAuthentictedUser, newOrder);

export const createOrder = (order) => async (dispatch) => {

  try {

    dispatch({ type: CREATE\_ORDER\_REQUEST });

    const config = { headers: { "Content-Type": "application/json" } };

    const { data } = await axios.post("/api/v1/order/new", order, config);

    dispatch({ type: CREATE\_ORDER\_SUCCESS, payload: data });

  } catch (error) {

    dispatch({ type: CREATE\_ORDER\_FAIL, payload: error.message });

  }

};

  async function paymentSubmitHandler(e) {

    e.preventDefault();

    if(nameOnCard === ""){

      alert.error("Please enter name on card");

      return;

    }

    try {

      const config = {

        headers: {

          "Content-Type": "application/json",

        },

      };

      const { data } = await axios.post(

        "/api/v1/payment/process",

        paymentData,

        config

      );

      const client\_secret = data.client\_secret;

      if (!stripe || !elements) return;

      const result = await stripe.confirmCardPayment(client\_secret, {

        payment\_method: {

          card: elements.getElement(CardNumberElement),

          billing\_details: {

            name: user.name,

            email: user.email,

            address: {

              line1: shippingInfo.address,

              state: shippingInfo.state,

              postal\_code: shippingInfo.pinCode,

              country: "IN",

            },

          },

        },

      });

      if (result.error) {

        alert.error(result.error.message);

      } else {

        if (result.paymentIntent.status === "succeeded") {

          order.paymentInfo = {

            id: result.paymentIntent.id,

            status: result.paymentIntent.status,

          };

          alert.success(result.paymentIntent.status);

          dispatch(createOrder(order));

          history.push("/success");

        } else {

          alert.error("There's some issue while processing payment");

        }

      }

    } catch (error) {

      alert.error(error.message);

    }

  }

const { shippingInfo, cartItems } = useSelector((state) => state.cart);

  const subTotal = cartItems.reduce((acc, currItem) => {

    return acc + currItem.quantity \* currItem.price;

  }, 0);

  const totalFinalPrice = subTotal;

  const order = {

    shippingInfo,

    orderItems: cartItems,

    itemsPrice: subTotal,

    shippingPrice: 0,

    totalPrice: totalFinalPrice,

  };

exports.isAuthentictedUser = asyncWrapper(async (req , res , next) =>{

    const { token } = req.cookies;

if(!token){

    return next(new ErrorHandler("Please Login to access this resource", 401));

}

    const deCodeToken = jwt.verify(token, process.env.JWT\_SECRET);

    const user = await userModel.findById(deCodeToken.id);

    req.user = user; // now we have user in req.user

    next();

})