Video 12:

**Navbar.js**

Import Badge from ‘react-bootstrap/Badge’;

….

…..

            {(!localStorage.getItem('authToken')) ?

              <div className="d-flex">

                <Link className="nav-link mx-2 fw-bold" to="/login">Login</Link>

                <Link className="nav-link mx-2 fw-bold" to="/signup">SignUp</Link>

              </div>

              :

              <div>

                <div className="btn bg-white text-success mx-2 fw-bold" onClick={()=>{setCartView(true)}}>

                  My Cart {"  "}

                <Badge pill bg="danger"> 2</Badge>

                </div>

                {

                  cartView ? <Modal onClose={()=>setCartView(false)}> <Cart/> </Modal>  : null

                }

              <div className="btn bg-white text-danger mx-2 fw-bold" onClick={handleLogout}>Logout </div>

              </div>

            }

                <div className="btn bg-white text-success mx-2 fw-bold" onClick={()=>{setCartView(true)}}>

                  My Cart {"  "}

                <Badge pill bg="danger"> 2</Badge>

                </div>

Add badge like this

Static badge – shows only 2 badge in cart button

To add space between cart and 2 badge added space as mycart{“ ”}

**Creating a component:**

**Cart.js**

Export default function Cart(){

Return(

<div>

<div className=’container m-auto mt-5 table-responsive table-responsive-sm table-responsive-md’

  return (

    <div>

      {console.log(data)}

      <div className="container m-auto mt-5 table-responsive  table-responsive-sm table-responsive-md">

        <table className="table table-hover ">

          <thead className=" text-success fs-4">

            <tr>

              <th scope="col">#</th>

              <th scope="col">Name</th>

              <th scope="col">Quantity</th>

              <th scope="col">Option</th>

              <th scope="col">Amount</th>

              <th scope="col"></th>

            </tr>

          </thead>

Addingcondition if cart is empty displaying as cart is empty

We used dispatch and state to send data to any component

Export useCart,useDispatchcart

We have add case in switch

So importing useCart and useDispatchCart in cart.js

So displaying as cart is empty when data.length==0

  if (data.length === 0) {

    return (

      <div>

        <div className="m-5 w-100 text-center fs-3 text-light !important">

          The Cart is Empty!

        </div>

      </div>

    );

  }

If this case is false it goes to next line

In this done data.map (food,index)

Data is type of array so data.length will works

Data means state imported using useCart

Totalprice calcualated using reduce method

  let totalprice = data.reduce((total, food) => total + food.price, 0);

  return (

    <div>

      {console.log(data)}

      <div className="container m-auto mt-5 table-responsive  table-responsive-sm table-responsive-md">

        <table className="table table-hover ">

          <thead className=" text-success fs-4">

            <tr>

              <th scope="col">#</th>

              <th scope="col">Name</th>

              <th scope="col">Quantity</th>

              <th scope="col">Option</th>

              <th scope="col">Amount</th>

              <th scope="col"></th>

            </tr>

          </thead>

          <tbody>

            {data.map((food, index) => (

              <tr>

                <th scope="row">{index + 1}</th>

                <td>{food.name}</td>

                <td>{food.qty}</td>

                <td>{food.size}</td>

                <td>{food.price}</td>

                <td>

                  <button

                    type="button"

                    className="btn p-0"

                    onClick={() => {

                      dispatch({ type: "REMOVE", index: index });

                    }}

                  >

                    <img

                      className=" img1"

                      src={trash}

                      alt="delete"

                    />

                  </button>

                </td>

              </tr>

            ))}

          </tbody>

        </table>

        <div>

          <h1 className="fs-2 text-light !important">

            Total Price: {totalprice}/-

          </h1>

        </div>

Trash img download and import .

**Creating modal**

Now lets say I want to display a webpage on top of that page . create a modal

On clicking mycart button a model should gets displayed

Src>Modal.js

Create Modal.js inside src folder

Everything we are rendering on only one div with id=”root”

For modal which is not a other separate website, it gets displayed infront of webpage. So the back webpage will be scrollable and changes done on that page will not get losed

Creating another div in index.html with id cart-root

    <div id="root"></div>

    <div id="cart-root"></div>

**Src>Model.js:**

import React from 'react'

import ReactDom from 'react-dom'

const MODAL\_STYLES={

    position: 'fixed',

    top: '50%',

    left: '50%',

    backgroundColor: 'rgb(34,34,34)',

    transform: 'translate(-50%, -50%)',

    zIndex: 1000,

    height:'90%',

    width: '90%'

}

const OVERLAY\_STYLES={

    position: 'fixed',

    top:0,

    left:0,

    right:0,

    bottom:0,

    backgroundColor: 'rgba(0,0,0, .7)',

    zIndex: 1000

}

export default  function Modal({children, onClose}){

    return ReactDom.createPortal(

        <>

        <div style={OVERLAY\_STYLES}>

            <div style={MODAL\_STYLES}>

                <button className='btn bg-danger fs-4' style={{ marginLeft: '90%' , marginTop: "-35px"}} onClick={onClose}>X</button>

                {children}

            </div>

        </div>

        </>,

        document.getElementById('cart-root')

    )

}

**Creating a state to dynamically display badge according to the items present in cart/data/state**

export const Navbar = () => {

  const [cartView,setCartView]=useState(false);

**initially setting cartview value as false**

**then on clicking mycart –cartview should be set to true and modal should be rendered**

**adding condition to render modal:**

                <div className="btn bg-white text-success mx-2 fw-bold" onClick={()=>{setCartView(true)}}>

                  My Cart {"  "}

                <Badge pill bg="danger" badgeContent={data.length}> {data.length} </Badge>

                </div>

                {

                  cartView ? <Modal onClose={()=>setCartView(false)}> <Cart/> </Modal>  : null

                }

Modal is sending a onclick{onclose()} props is sending from Models.js on clicking close button(x)

So here rendering model when cartview is set to true, and passing onclose function

Set to false to hide

Pasing cart inside modal

<Modal onClose={()=>setCartView(false)}> <Cart/> </Modal>

  cartView ? <Modal onClose={()=>setCartView(false)}> <Cart/> </Modal>  : null

this cart component will get accessed in Modal.js using {children}

now check output by adding some elements into cart and click mycart button.

On reload data is getting lossed . added cart data. Bcoz we are not storing state data in local storage or any storage.

We are just storing in state var, which is not reliable, in real-time,

In real-time , data must be stored in local storage or server

**Creating badge dynamically:**

Badge must display total number of elements present in cart, instead of just a static 2 value.

So import data from useCart() , state

export const Navbar = () => {

  const [cartView,setCartView]=useState(false);

  const data=useCart();

                <Badge pill bg="danger" badgeContent={data.length}> {data.length} </Badge>

Data.length instead of 2

**Creating delete or remove functionality:**

ContextReducer.jsx

Add remove case in switch

        case "REMOVE":

            let newArr=[...state];

            newArr.splice(action.index,1)

            return newArr

just storing whole state data in one var newArr

and splicing by one – removing one

without changing original state, assigned and changed newArr.

Check array method splice

Then on returning new arr, cart.js will rerender as data=useCart()

And in cart.js we have remove button to delete

In that button call dispatch function with type: remove

                  <button

                    type="button"

                    className="btn p-0"

                    onClick={() => {

                      dispatch({ type: "REMOVE", index: index });

                    }}

                  >

                    <img

                      className=" img1"

                      src={trash}

                      alt="delete"

                    />

                  </button>

Now check on which element we are clicking only that element is getting deleted.

Add and remove functionality will be similar for all applications

Update- is different will be specific to that particular application

**Update functionlaity:**

Card.js

     const handleAddToCart=async()=>{

        let food=[]

        for(const item of data){

            if(item.id === props.foodItem.\_id){

                food=item;

                break;

            }

        }

        console.log(`food is ${food}`)

food is any empty array

item.id means the id present in state(data) is equal to props.fooditem.id

i.e., comparing item.id with the item which we add to cart just now

if it already exits then food=item;

else we are breaking

break –will come out of for loop

now using food we can update

update logic will depend on food here

this above for loop will check whether that fooditem.id already exists or not in state(data)

if condition will get true only when that item already exits, else comes out of loop

now if(food!=[]) if food is not equal to empty array then check if size is similar and only qty is updated then do update.

i.e.,

eg:

1. Chicken noodles = qty(1) , size(half)
2. Chicken noodels = qty(2) , size(half)

So here size is same , only qty changed so update

Eg:

1. Chicken noodles = qty(1) , size(half)
2. Chicken noodels = qty(2) , size(full)

So here size also changes so instead of update, change

As both are different orders – add

Only qty changing – update

     const handleAddToCart=async()=>{

        let food=[]

        for(const item of data){

            if(item.id === props.foodItem.\_id){

                food=item;

                break;

            }

        }

        console.log(`food is ${food}`)

        if(food != []){

            if(food.size === size) {

                await dispatch({type:"UPDATE", id: props.foodItem.\_id,price:finalPrice, qty:qty})

                return

            }

            else if(food.size!== size){

                await dispatch({type:"ADD", id:props.foodItem.\_id,name:props.foodItem.name,price:finalPrice,qty:qty,size:size})

                console.log("Size different so simply ADD one more to the list")

                return

            }

            return

        }

        await dispatch({type:"ADD",id:props.foodItem.\_id,name:props.foodItem.name,price:finalPrice,qty:qty,size:size});

        await  console.log(data)

        await console.log('handleaddtocart working')

     }

If foodsize===size and foodsize!==size both conditions are not true , then also simply adds

Add return statement after every operation

**Defining update case in switch:**

        case "UPDATE":

            console.log('update running')

            let arr=[...state]

            arr.find((food,index)=>{

                if(food.id === action.id){

                    console.log(food.qty,parseInt(action.qty),action.price + food.price)

                    arr[index]={...food, qty: parseInt(action.qty)+food.qty, price:action.price + food.price}

                }

                return arr

            })

            return arr

**taken state into a temp var**

**let arr=[…state]**

**then applied find operation on it**

**arr.find(food,index)**

**index we are sending already**

**food.id means which already exists in state that id and action.id means which we sent while clicking add to cart**

**if(food.id === action.id)**

**qty converting to int parseint(action.qty) which we got on clicking add to cart + food.qty which is already existing in state data.**

**Similarly price: action.price + food.price**

**Check now by adding**

**Video 13: checkout functionality:**

**When clicking checkout button in cart.js , our whole data with which user has added to cart that particular user data attached with the items should get stored into db.**

**For one particular user, his item details,**

**For another user, other item his own item details must be stored**

**For this purpose we need 2 things, one separate collection to store data**

**And the other thing is its endpoint should be created in backend**

**In users collection in mongodb atlas, users email,passwords are getting stored whole user data**

**Create another collection order, in that users.email should also be a part of order collection to identify which data is of which user**

**In models folder, create file Orders.js**

**Creating schema :**

**const mongoose=require(‘mongoose’);**

**const { Schema } = mongoose;**

const mongoose=require('mongoose');

const {Schema }=mongoose;

const OrderSchema=new Schema({

    email:{

        type:String,

        required:true,

        unique:true

    },

    order\_data:{

        type:Array,

        required:true,

    },

})

module.exports=mongoose.model('orders',OrderSchema)

**creating a route**

**Routes >OrderData.js**

**Const express=require(‘express’)**

**Const router=express.Router();**

**Const Order=require(‘../models/Orders’);**

**Now we have to understand how to send data from cart.js to OrderData.js**

**So that to keep that data into schema and sending it to db orders collection**

**So when clicing checkout button (which is present in cart.js)- we have to send data (state) and that data must be associated with the user who logined.**

**Eg: when shamdas was loginned , then shamdas data should only be send**

**So associated email should be send along with data**

**Now check login.js**

**Here on login , if(json.success) we can store email in localstorage**

    if (json.success) {

      localStorage.setItem('userEmail',credentials.email);

      localStorage.setItem('authToken',json.authToken);

      console.log(localStorage.getItem('authToken'))

      navigate('/');

    }

**Our use email is saved in credentials.email**

**So storing credentials.email into localstorage with key name “userEmail’**

      localStorage.setItem('userEmail',credentials.email);

**this saved userEmail only we can send it when ever we click checkout button**

**for this we need to implement one fetch api in cart.js**

**so for fetching , we have to create a endpoint in backend**

**creating endpoint in routes>OrderData.js**

and apart from data, from cart.js , we can send date also

creating endpoint – method post, endpoint