**How to use react-router**

Changes the product model from the API side are as follows firstly size and color attributes are now array as they store array of objects in each case and added new attribute inStock in Boolean.

We also need to add some path that is we need to route the components accordingly so for that we need react-router-dom library and for that we need Router- Switch—Route (this type of ancestry and for further information read the documentation).

In this unit we basically added various routes for various pages and added links to the names of categy to redirect them to category page.

**How to fetch and filter products**

Now to sort the the fetched products firstly we need to head to the product list page from where the all the products are being rendered and we need to pass the properties such as categories, color,size and price sort.

We can get the categories from the url using uselocation hook and other properties from the assign them to a state variable using the onChange method in the container.

So now if we have the required values of the filters then we can use them for filtering using useEffect.

First if any category is given the we can request the api route with the category querry and for the second useEffect we can use the filters(props) to filter the prev returned array from api request and for the last useEffect that is the price sorting we sort the array with respect to different aspect.

**How to fetch a single Product**

To fetch a single product data firstly we need link the serch icon on the products(when we hover over them) to the single product page.

So to get a product we just need it id and then we can just use the api find/:id route to get that product.

We also use some state variable to account for the quantity size color selected so there that

In this section we have also created a generic requestMethods.js file which contains public request and userRequest then created by user also had Bearer token so need to generate from login.

**Redux toolkit shopping cart**

Better understanding of redux watch it dedicated video on it its state management tool.

Now the add to shopiing cart functionality is done through redux as we want to change in state of it intantly as we click the add to cart button so for that we install @reduxjs/toolkit and react-redux.

Firstly we create a redux folder where we create a state for the cart it will have products as array and quantity , and price initial as empty and zero rept.but if there is a change in state we increase it quanity and other attributes.

Now since this is out of the way we can move on to the products page where we use another hook called useDispatch with the help of which we could change the state of the cart in the navbar.

We use useSelector in the navbar to select the specific state variable.

**React shopping cart**

Firstly we link the cart icon to the cart route.

To list the items in shopping cart we just use the state of cart so with the help of useSelector we get the state of the cart and then we map the products array in the cart page with all the desired attributes.

**React Stripe Payment tutorial**

Basically used the node api route and the documentation to complete the post request and also having some difficulties in giving feedback to the user in successful transaction.

Rewatch the node api and react part of it bit complex in its own way.

**React Redux toolkit authentication**

So similar to the cart implementation we also need to always need to have a state for the user logged in so for that now we need to create a redux state.

Initially currentUser will be null , isFetching also be false and there would be no errors . but in loginStart fetching will be true and in success fetching will be false (stop) and no errors and currentUser will be equal to the payload and in the failure case we have isFetching =false and error true…typical state changes

Although we can make api calls in the login page itself but for the cleanliness I tend to create a api calls file which gives api calls initially loginStart🡪 axios call 🡪 success else failure

And now in the login file we just create some state variables and then use the apiCalls’s login method and pass the desired parameters as arguments to get the login request but with this there is a problem if we reload the page the user in currentUser will revert back to null so we have to login again which is not good.

**Redux toolkit Persist Tutorial**

Read some docs not able to understand fully but it was doc based implementation.

**React E-commerce admin**

So for the admin panel we are using the prev UI created but we are also giving it another login page with only username and password fields and on submit we will give it call to backend so we need to put all the api calls in the apiCalls.js redux folder.

We created some redux files similar to that of client side like userRedux store and apiCalls and requestMethods. Instead of giving user a static token we give user a dynamic token using the persist

const TOKEN = JSON.parse(JSON.parse(localStorage.getItem("persist:root")).user).currentUser.accessToken;

so through this we can get token of the user from local storage.

**Fetching last n elements from the api**

In this section we fetch users and orders from the node api query and show them in small and large widget respectively.

const res = await userRequest.get("users/?new=true"); with the help of this querry we can get the latest 5 users so this is how we get user in the small widget.

const res = await userRequest.get("orders"); Similarly this is for the orders/ transactions.

**React.js Chart Tutorial**

We remember that when we get the user stats from the node api it returns us month no and total no of active users so now our task is to convert that number to correspond to a month .

To correspond the number to a month we use a array method where we are storing month name in an array and then mapping app the element the come as response on the get request of the user stats

res.data.map((item) =>

          setUserStats((prev) => [

            ...prev,

            { name: MONTHS[item.\_id - 1], "Active User": item.total },

          ])

Here Month is the array and item.id gives the month number and since the array is zero indices so we use item.id-1. And then we use user stats as data frame which is an array.

**React.js Calculating Revenue**

To calculate the revenue we are using the orders/income route with gives the total expenditure in the array format with prev month and last to last month record.

So now to show the data we can just put them in state variables and render them accordingly.

For the percent increase and decrease we just find the ratio of new to prev total earnings.

**Redux Toolkit CRUD operations**

To fetch the products from the db we need to create a product slice of productRedux for it whose initial state is just like user empty array isfetching is false and error is false.

So getProductStart isfetching is true ,error is false if success we have action payload so we assign this payload to products array and is fetching is false else if it is a failure then ifFetching is false ,error is true.

Add this reducer to store and create a similar function to that of login in apiCalls.js.

So this was for the backend now to use this setup for fetching of data we use useDispatch and useSelector and useEffect to dispatch our data to the server as soon as we render the page.

Now with the help of useSelector we can access the state of product and get the products and put them in ui.

Delete functionality also similar to that of create we make reducers and then for delete we need is only id so we get it with the api call and then with the help of splice function we remove the that element from the products array.

And we invoke the handledelete where we use dispatch to delete in delete icon in UI.

Now to work on the specific product when we click on the product we want to show its features and make some important changes.for that when we click on the edit we need the params id for that specific product so to do that we can use useLocation hook.Now with the help of the pathname given by the useLocation hook we can find that specific product in the products array whose id is is required.

Now moving on to the chart on the singular product page(stats of product) to do that we make some updates in our api which also give data matching with the product it when qurried with it.

        const res = await userRequest.get("orders/income?pid=" + productId);

now we sort the data with respect to newest to oldest

        const list = res.data.sort((a,b)=>{

            return a.\_id - b.\_id

        })

// now we map this data to that of the stats for the chart

        list.map((item) =>

          setPStats((prev) => [

            ...prev,

            { name: MONTHS[item.\_id - 1], Sales: item.total },

          ])

        );

Now we add reducers for the final update and create operation all the start and failure reducers are some but for success in update we change the value at the particular index of the given id element and in case of create we just push the payload in the array.

**Handling multiple input in newProduct page**

In the new product page we have multiple inputs to manage so here one could use multiple states and in the end combine them in the end to an object.

But here we use only one state for all input except categories(as this is array) and image(as we need to upload it somewhere and then get the url).

So we put names and handleChange onChange functions and manage their state and for the categories we need only the values separated by commas so we just change their values by spliting them from commas.

  const handleChange = (e) => {

    setInputs((prev) => {

      return { ...prev, [e.target.name]: e.target.value };

    });

  };

  const handleCat = (e) => {

    setCat(e.target.value.split(","));

  };