Game Console Ecommerce Website.

Project to understand React Routes & use of Internal data & External data (API)

Step 1

Create a React App with npx create-react-app. Install react router with npm install react-router-dom. Install Bootstrap with npm i bootstrap@5.3.0-alpha3

Now if go to package.json in our project files, we can see now bootstrap and react-router-dom is installed in our dependencies.

Step 2

Now we will remove everything from App.js which includes logo imports and all of css stylesheet codes of App.css and we will change title of our webpage in our index.html which is under folder public.
Now in App.css we will change background color and text color of our document.

(in App.css)

```
body {
background-color: rgb(28, 28, 28);
color: white;
}
```

Step 3

First we will set us our routes for pages in our website.
So in App.js we will import react-router-dom

```
import { BrowserRouter, Routes, Route } from 'react-router-dom';
```

Now we will use <BrowserRouter>.
A `<BrowserRouter>` stores the current location in the browser's address bar using clean URLs and navigates using the browser's built-in history stack.
It is the parent component used to store all other components.
So we have used BrowserRouter as a router to of App.js file to wrap all other components. BrowserRouter is a parent component and can have only one child.

<Routes> are perhaps the most important part of a React Router app. They couple URL segments to components.
Inside <Routes> we have <Route>.
<Route> is used to define and render component based on the specified path. It will accept components and render to define what should be rendered

```
<BrowserRouter>
<Header />
<Routes>
<Route path='/' element={<Home />} />
<Route path='about' element={<About />} />
<Route path='products' element={<Products />} />
<Route path='products/:productId' element={<SingleProduct />} />
<Route path='communitytalks' element={<Posts />} />
<Route path='*' element={<Error />} />
</Routes>
<Footer />
</BrowserRouter>
```

So Here, we will have <header> and <footer> which will component to be displayed. So do not require to change header and footer when user browse our website. We only will render the components such as home, posts and products between header and footer of our website to change. For this, we have used

<Routes> to handle all components .we have created several path and component to rendered by using <Route>. So we will be creating several components as required. We have used “ * ” path to handle the error if user enters undefined pathname in URL for website. So we will also create error component that will be displayed to handle undefined path.

It is important to note we have used path products/:productId
The syntax ":productId" is used in the context of React Router to create a dynamic route parameter. It is used to match and extract a parameter value from the URL path.
":productId" specifies a placeholder for the actual product ID, which can be any string value. When the user visits a URL containing a product ID, React Router can extract the ID from the URL path and use it to retrieve and display the corresponding product page.

Step 4

We will create a functional component <Header> and <Footer> components as required. So for this we will create folder under src folder of our project file and name it as components, we will create two file in this folder, Header.js & Footer.js .

Now we will create another folder under src folder of our project file and name it as pages. We will create all component we need and that we defined in our Route paths.
So we will create a component to render for home page, about page, product page, products details page, community talks page and error page

Now we will simply import all components in our App.js

```
import Header from './components/Header';
import Footer from './components/Footer';

import About from './pages/About';
import Error from './pages/Error';
import Home from './pages/Home';
import Posts from './pages/Post';
import Products from './pages/Products';
import SingleProduct from './pages/SingleProduct';
```

Step 5

Now we will create our Header Component using Bootstrap.
In order to use bootstrap, we need to import it first. So we will import bootstrap dependencies that we installed in our project earlier on our Step 1 in our App.js

(in App.js)

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Now we will use Bootstrap to create Header.js

(in Header.js)

```
<nav class="navbar navbar-expand-lg bg-dark" data-bs-theme="dark">
<div class="container-fluid">
<a class="navbar-brand" href="#">GamingConsole.com</a>
<ul class="navbar-nav float-end">
<li class="nav-item">
<a class="nav-link active" aria-current="page" href="#">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">About</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Product</a>
</li>
<li class="nav-item">
<a class="nav-link" href="#">Community Talks</a>
</li>
</ul>
</div>
</nav>
```

Now we will change all anchor tags to Link. <Link> is an react-router element. A `<Link>` is an element that lets the user navigate to another page by clicking or tapping on it. In `react-router-dom`, a `<Link>` renders an accessible `<a>` element with a real `href` that points to the resource it's linking to. This means that things like right-clicking a `<Link>` work as you'd expect.

So we will use <Link> instead of <a> anchor tag and rather than href attribute we will to attribute to define its path.

So first we will import Link from React Router Dom.

```
import {Link} from 'react-router-dom';
```

Now we will set our <Link> element

```
<nav class="navbar navbar-expand-lg bg-dark" data-bs-theme="dark">
<div class="container-fluid">
<Link to='/' class="navbar-brand" >GamingConsole.com</Link>
<ul class="navbar-nav float-end">
<li class="nav-item">
<Link to='/' class="nav-link active" aria-current="page" >Home</Link>
</li>
<li class="nav-item">
<Link to='about' class="nav-link" >About</Link>
</li>
<li class="nav-item">
<Link to='products' class="nav-link" >Products</Link>
</li>
<li class="nav-item">
<Link to='communitytalks' class="nav-link" >Community Talks</Link>
</li>
</ul>
</div>
</nav>
```

So Now if we click on our Navbar in our webpage, we can see components getting rendered.

Step 6

Now we will create Footer Component with Bootstrap.
For this, we will first we will get FontAwesome accessible to get icons available to us to use it for design purpose.

We will go public folder in our project and we will open index.html

we will add CDN of FontAwesome inside <head> element of this html file.

```
<link rel="stylesheet" type="text/css"
href="//use.fontawesome.com/releases/v5.7.2/css/all.css">
```

Now we will create a footer of our webpage with icons in Footer.js
we will also import Links from React-Router-Dom to link homepage with url address on bottom of footer of our website.

```
<footer class="bg-dark text-center text-white">
<div class="container p-4 pb-0">
<section class="mb-4">
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-facebook-f"></i
></a>
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-twitter"></i
></a>
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-google"></i
></a>
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-instagram"></i
></a>
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-linkedin-in"></i
></a>
<a class="btn btn-outline-light btn-floating m-1" href="#!" role="button"
><i class="fab fa-github"></i
></a>
</section>
</div>
<div class="text-center p-3 bg-dark opacity-75">
© 2020 Copyright:
<Link class="text-white" to='/'>GamingConsole.com</Link>
</div>
</footer>
```

Step 7

Now we will create a Home Component. We will have this component to act as homepage.
So we require background image for this and we cannot use CSS inline attribute, we need to create an object which defines the css styling of element and use this object of key-value pair to define the inline styling.

```
const myStyle = {
backgroundImage: "url('https://wallpaperaccess.com/full/4848691.jpg')",
height : "100vh",
backgroundRepeat: "no-repeat",
backgroundPosition : "center",
};
```

We will also use React Router to link the buttons to different components of webpage.

```
import { Link } from 'react-router-dom'
```

We have now our HomePage as required .

```
<div
id="intro-example"
class="p-5 text-center bg-image"
style={myStyle} >
<div class="mask" >
<div class="d-flex justify-content-center align-items-center h-100">
<div class="text-white">
<h1 class="mb-3">Welcome to World of Game Consoles</h1>
<h5 class="mb-4">Best & Free guide for your Daily Gaming Assistance </h5>
<Link
class="btn btn-outline-light btn-lg m-2"
to='products'
>Go to our Store</Link>
<Link
class="btn btn-outline-light btn-lg m-2"
to='communitytalks'
>Talk to other Gamers</Link>
</div>
</div>
</div>
```

```
</div>
```

Step 8

Now we will be created our About component. We will use html and css to design as we require.

We will be creating a new file About.css which will contain all necessory css stylesheet required to design our webpage and import this file in our About.js

Step 9

Now we will create a database which will carry all products and its details to use. We will create a js file structure of array with objects items where each object will have key-value pair describing the details of products.

So we will create Data.js

this will contain a array of object we will be using to create product display. We will create this file in our src folder of our project.

We require product images so we will store all our images inside a new folder in our src folder. We will name it as images.

```
const PRODUCTS = [
{
id: 1,
name: 'Xbox ONE X',
price: '$999',
image: require('./images/1.jpeg'),
details:
'Immerse yourself in the world of fantasy with the true 4K gaming experience provided by the Xbox One X. Whether you are shooting your way past enemies in Gears of War 4 or blazing through the streets in Need for Speed Payback, this console brings the action to life, thanks to the brilliant graphic details that are enhanced by the High Dynamic Range technology.',
},
{
id: 2,
name: 'Nintendo GameBoy',
price: '$349',
image: require('./images/2.jpeg'),
details:
' Built-in 400 classic games Can be played on TV Rechargeable Lithium-ion battery and USB cable, 6 hours of continuous game play after full charge Small size and light weight, suitable for traveling Perfect gift for children, retro game lovers, collectors, etc.'
},
{
id: 3,
name: 'PlayStation Remote',
price: '$299',
image: require('./images/3.jpg'),
details:
'The DualShock (originally known as Dual Shock; trademarked as DUALSHOCK or DUAL
SHOCK; with the PlayStation 5 version named DualSense) is a line of gamepads with
vibration-feedback and analog controls developed by Sony Interactive Entertainment for
the PlayStation family of systems.',
},
{
id: 4,
name: 'xBox Wireless Remote',
price: '$497',
image: require('./images/4.jpg'),
details:
'The Xbox Wireless Controller is the primary game controller for the Xbox One and Xbox
Series X/S home video game consoles, also commercialized for its use in Windows-based
PCs, and compatible with other operating systems such as macOS, Linux, iOS, and
Android.',
},
{
id: 5,
name: 'Insurance',
price: '$100.00',
image: require('./images/5.jpg'),
details:
'Get Insurance for any damage with our ConsoleInsurance',
},
{
id: 6,
name: 'Vintage Console Available',
price: '$799',
image: require('./images/6.jpeg'),
details:
'Get Your Childhood back with Old School Gaming Console ',
},
];

export default PRODUCTS;
```

So our variable PRODUCTS is a array of objects which contains id, name, price, image and details about our products. We will access to those object properties to display in our webpage when we require those content.

Step 10

Now we will create Product.js. So we will Bootstrap Cards to display all products.
We have always have to make remember to make code DRY (Don’t Repeat Yourself) so rather than Bootstrap Card for each and every item we will use map() method in our array which we run to execute a function for all items inside the array
This will allow us to access the Object inside the Array and use its properties such as name, price, image and details to display at required place in our Bootstrap Card.

First we will import PRODUCT variable we created from Data.js

```
import PRODUCTS from '../Data.js'
```

Now we will use map() method to run a function for each array’s item. We will design a function to create a Bootstrap Card and use Object’s properties to position the content accordingly.

```
<div className="container content d-flex justify-content-center mt-4">
<div className="row products-row me-2 mt-4">
{PRODUCTS.map( (product) => {
return (
<div className="col-lg-4 mt-4" key={product.id}>
<div className="card h-100 d-flex flex-column text-white bg-black">
<div className="img-wrap align-self-center">
<img src={product.image} alt="" />
</div>
<div className="card-body bg-black bg-gradient flex-grow-1">
<h5 className="card-title">{product.name}</h5>
<p className="card-text">{product.details}</p>
<div className="d-flex justify-content-between align-items-center">
<span>Price: <strong className="price">{product.price}</strong></span>
</div>
</div>
</div>

</div>
```

Step 11

We want a Button inside our Bootstrap card that renders the product page. This page will contain images and content based on which button of Bootstrap card clicked.

So we will import first from react router dom and add <Link> to render <SingleProduct /> element which has a path to products/:productId

As Mentioned Earlier we have used :productId to create dynamic routes.

```
<Link to={`/products/${product.id}`} className="btn btn-primary btn-sm">DETAILS
&#8594;</Link>
```

So Now our Product.js will be coded as,

```
import { Link } from 'react-router-dom'
import PRODUCTS from '../Data.js'
const Products = () => {
const myStyle = {
backgroundColor:"rgb(28, 28, 28)",
};
return (
<main style={myStyle}>
<div className="pg-header bg-black text-white align-items-center ">
<div className="container">
<div className="text-center">
<h1>Products & Services</h1>
</div>
</div>
</div>
<div className="container content d-flex justify-content-center mt-4">
<div className="row products-row me-2 mt-4">
{PRODUCTS.map( (product) => {
return (
<div className="col-lg-4 mt-4" key={product.id}>
<div className="card h-100 d-flex flex-column text-white bg-black">
<div className="img-wrap align-self-center">
<img src={product.image} alt="" />
</div>
<div className="card-body bg-black bg-gradient flex-grow-1">
<h5 className="card-title">{product.name}</h5>
<p className="card-text">{product.details}</p>
<div className="d-flex justify-content-between align-items-center">
<span>Price: <strong className="price">{product.price}</strong></span>
<Link to={`/products/${product.id}`} className="btn btn-primary btn-sm">DETAILS
&#8594;</Link>
</div>
</div>
</div>
</div>
)
} )}
</div>
</div>
</main>
)
}

export default Products
```

Step 12

Now we will create a single product page which is <SingleProduct /> component.
In App.js we have a ":productId" that specifies a placeholder for the actual product ID.

(in App.js)

```
<Route path='products/:productId' element={<SingleProduct />} />
```

Now we will get this :productId in our SingleProduct.js
So we will have useParams() hook.
This allow us to access the parameters of the current route.
So first we will import useParams from react

(in SingleProduct.js)

```
import { useParams } from 'react-router-dom'
```

```
const { productId } = useParams();
```

Here we have retrieved productID variable from the URL with help of useParams hook's returned object and stored in ProductId constant.

Now it is important to note that if we use console.log(typeof(product.Id)) we get this as string.

Now we need to find object from Data.js based on its id.
So first we will import PRODUCTS from Data.js

```
import PRODUCTS from '../data.js'
```

Now we can have a variable to store that particular object from our Data.js that has its ID equal to ID we retrived from URL.

```
const singleProduct = PRODUCTS.find(product => product.id === parseInt(productId))
```

So this will find a single object from our PRODUCTS. We have set the condition that id property of each product object must be equal to interger value of productId. We have converted string to interger with parseInt() function. Once the object is found inside this PRODUCT array, it will store itself in singleproduct variable.

We will destructure object we found to extract the values of properties from this object. This is just so it is easy to code. We could have used singleProduct.name,singleProduct.price etc... but this makes it easier to write.

```
const { id, name, price, image, details } = singleProduct
```

we can console.log(“product found” , singleProduct) to check we recieve our targeted object even if we change id from url.

Now we will design and set the value from object as required.

```
<main className="bg-black text-white">
<div className="pg-header bg-black text-white text-center">
<div className="container">
<div className="row align-items-center">
<div className="col-lg-7">
<h1>{name}</h1>
</div>
</div>
</div>
</div>
<div className="container content">
<div className="row">
<div className="col-lg-5">
<img src={image} alt="" className="img-fluid" />
</div>
<div className="col-lg-7">
<h2>{name}</h2>
<p className="price"><strong>{price}</strong></p>
<p>{details} {details} {details} {details} {details} </p>
</div>
</div>
</div>
</main>
```

Now we require back button which can take us to one step back from its current url so for this we can use navigate from react.

First we will create a button that has onClick which execute a navigation to move one step back.

```
<button className="btn btn-dark m-4 btn btn-outline-light " onClick={() => takeme(-1)}>More Products</button> &nbsp;
```

Now we will define this takeme.
takeme is just a variable which uses useNavigate() function.
So we will import useNavigate from react router dom

```
import { useNavigate , useParams } from 'react-router-dom'
```

Now

```
const takeme = useNavigate();
```

We have its parameter on click to be -1 which means it will take us one step back.

We will also create a link to take us to homepage. It is important to import this from react router dom.

```
<Link to="/" className="btn btn-dark btn-sm btn float-end ">Home</Link>
```

Step 13

Now we will Create a Community Talk page of <Post /> component .js. We will fetch external data to display on our website. So we will have API to get us our required data. We will set Data as we did in our Product.js with using data object of an array and its key-value pair.

So first we will use USESTATE hook. This will create a state variable. We will make its initial value to be an empty Array.

We will also create an another state variable using USESTATE hook. This will be of boolean value. We will have its initial value to be true. We will change its value to be false when we successfully recieved data from API.
We need this boolean state variable because we will run a conditional statement to handle the display if API is not successfully loaded.

So first we will import USESTATE hook from react

```
import { useEffect } from 'react';
```

Now we will have our state variables

```
const [ posts, setPosts ] = useState([]);
const [ loader, setLoader ] = useState(true);
```

We need a function to fetch our API data to our website. For this we will use USEEFFECT hook and call a function inside which allow us to fetch external data API to our webpage. USEFFECT hook allow us to create side effects in our component.

So we will import this hook first.

```
import { useEffect, useState } from 'react';
```

Now we will call a function inside useEffect hook which we will design to work with external API

```
useEffect(() => {
fetchPosts();
}, []);
```

Now it is important to note, we have use empty array. This is because we want to render the component only once when the browser loads.

Now we will define fetchPost() function to get us data from API.

We will define this function as asynchronous function. This means we want this function to return to use a promise.
Now inside this async function we will use fetch() method to get data from API. For this we will will use await. Await is very important keyword for our function as this will make a function wait for a promise.
We will go to https://jsonplaceholder.typicode.com/ and go to comment.
This will open a link that has a data like one we create in Data.js for PRODUCT variable array.
We will copy this link from our browser and use this to get us data for our website.

```
async function fetchPosts() {
await fetch('https://jsonplaceholder.typicode.com/comments')
}
```

Using the `await` keyword, the function waits for the promise returned by the `fetch()` method.
Now we will .then() which is promise method that allows us to handle the response of a fetch request.
`.then()` methods to the end of the call to handle the response returned by the server.

Now we will need to have the `json()` method to parse the response body as JSON.
The resulting JSON data is then passed to the `setPosts()` function which saves the data into a variable or state.

the `setLoader()` function is called with the boolean value `false` to indicate that the data fetching/loading is complete.

```
async function fetchPosts() {
await fetch('https://jsonplaceholder.typicode.com/posts')
.then(response => response.json())
.then(json => setPosts(json))
.then(setLoader(false))
}
```

We can console.log(posts) in our useEffect() to check the input of data from API. So now our posts variable which was value as an empty array now has external data.

Now we will have post state variable have the array of object data from our API. We will use this like we used before to create Product.js to access and set the content using map method()

```
(
<div className="content">
{loader
? <div className="text-center">Loading...</div>
: (
<div className="row">
{posts.map(post => {
return(
<div className='bg-black'>
<div className='d-flex flex-column mb-3 ' key={post.id}>
<div class="card text-center w-50 m-4 align-self-center">
<div class="card-header" style={bgcolor}>{post.email}</div>
<div class="card-body " style={bgstyle}>
<h5 class="card-title">{post.name}</h5>
<p class="card-text">{post.body} </p>
<a href="#" class="btn btn-dark">Reply</a>
</div>
<div class="card-footer" style={bgcolor}> Posted XX Days ago</div>
</div>
</div>
</div>
)
})}
</div>
)
}
</div>
)
```

We have use terniary operation to set display the text ‘loading...’ when data is not recieved from API. We have used our boolean state variable loader for this.

Hence our community talk page to display <Post /> component that has data from external API and we have also create a <Poduct /> component that has Data we created

Step 14

Now let us set the error page.
Error page is nothing a <Error /> component with a path = “*” .
This will handle any undefined path entered by user.

We will design it a single background image

```
const myStyle = {
backgroundImage: "url('https://media.istockphoto.com/id/1326959978/video/game-over-
text-animation-with-alpha-channe.jpg?
b=1&s=640x640&k=20&c=kaJY1x1p9cIEYFnrznFoy2sSkQyqrUbdYAO4ATgay_U=')",
height : "100vh",
backgroundRepeat: "no-repeat",
backgroundPosition : "center",
backgroundColor: "black",
};
```

```
return (
<div>

<div className='' style={myStyle}>

</div>
```

Step 15

Now if we look at out Navigation Bar at the top, we need change css styling of active link. We want to highlight the current active link.
For this, we will go to header.js and instead of <Link> we will use <NavLink>. Both <Link> and <NavLink> are components provided by react router dom to create links for routing. Link component is used simply to create links while NavLinks can provide additional feature to style and highlight active links based on URL.

So we will change all <Link> components in our header.js to <NavLink>.
We will first import it from react router dom

```
import { Link, NavLink } from 'react-router-dom';
```

Now we will define bootstrap class in class to condition.
If current link is actitive, we will have class to be “nav-link active” and when it is not active we will just have class to be “nav-link”

```
className={(navData) => navData.isActive ? 'nav-link
active' : 'nav-link'}
```

Now we will use it for every <Link> components.

```
<nav class="navbar navbar-expand-lg bg-dark" data-bs-
theme="dark">
<div class="container-fluid">
<Link to='/' class="navbar-brand"
>GamingConsole.com</Link>
<ul class="navbar-nav float-end">
<li class="nav-item">
<NavLink to='/' className={(navData) => navData.isActive ?
'nav-link active' : 'nav-link'}>Home</NavLink>
</li>
```

```
<li class="nav-item">
<NavLink to='about' className={(navData) =>
navData.isActive ? 'nav-link active' : 'nav-
link'}>About</NavLink>
</li>
<li class="nav-item">
<NavLink to='products' className={(navData) =>
navData.isActive ? 'nav-link active' : 'nav-
link'}>Products</NavLink>
</li>
<li class="nav-item">
<NavLink to='communitytalks' className={(navData) =>
navData.isActive ? 'nav-link active' : 'nav-link'}>Community
Talks</NavLink>
</li>
</ul>
</div>
</nav>
```

Now if we can see our Navigation bar links works perfectly.

Step 15

Now our website is complete. If we go to reactrouter.com it says an <outlet> should be used in parent router elements to render their child elements.

So for this, we will create a new component, MainLayout.js

This will be parent element of our website. So it will have Header and Footer components and <outlet /> component in between which will render child elements.

```
import { Outlet } from 'react-router-dom'

import Header from '../components/Header.jsx'
import Footer from '../components/Footer.jsx'

import React from 'react'

export default function MainLayout() {
return (
<>
<Header />
<Outlet />
<Footer />
</>
)
}
```

Now in App.js, we will import this MainLayout.js component.
We dont require <Header /> and <Footer /> now but rather use <Route> to have <MainLayout> element and wrap it around all its child elements.

```
<BrowserRouter>
<Routes>
<Route path='/' element={<MainLayout />}>
<Route index element={<Home />} />
<Route path='about' element={<About />} />
<Route path='products' element={<Products />} />
<Route path='products/:productId' element={<SingleProduct />} />
<Route path='posts' element={<Posts />} />
<Route path='*' element={<Error />} />
</Route>
</Routes>
</BrowserRouter>
```