

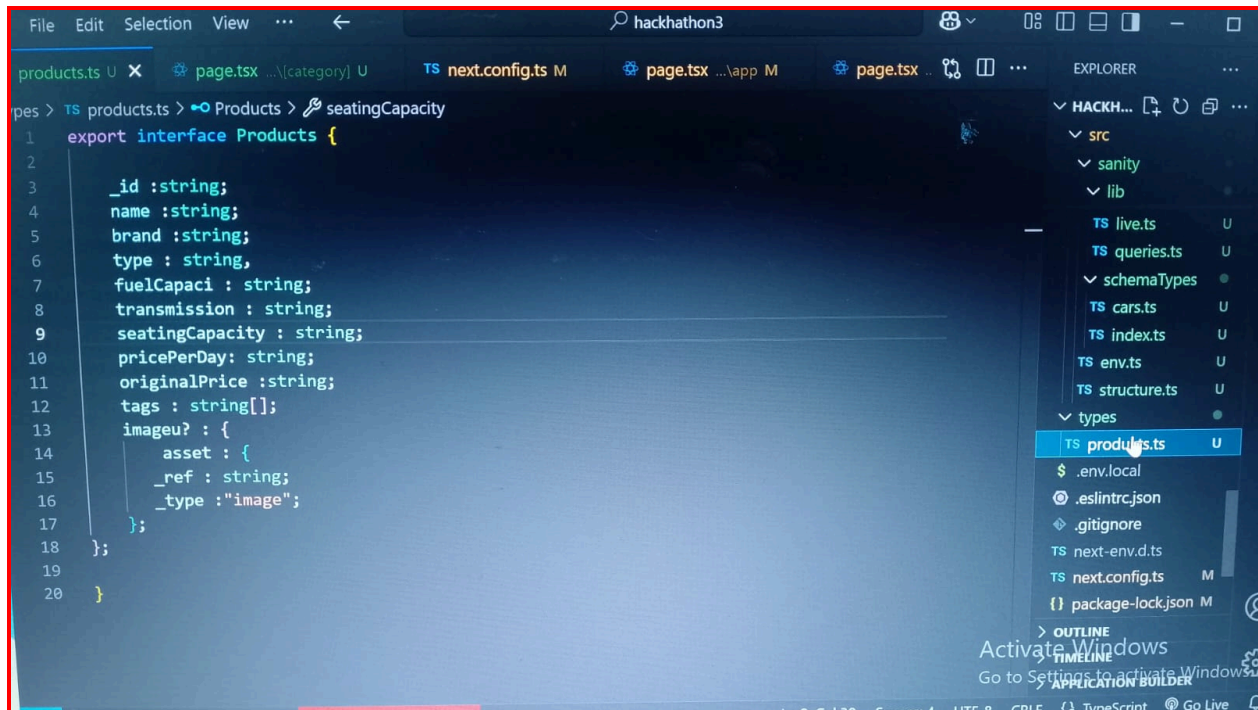
Dynamic /functionality Coding in multiples pages.

Dynamic /functionality Coding in rental car website:

```
src > app > categories > page.tsx > CategoriesPage
1 // This is my Categories page , Made by Aziza siddiqui //
2 'use client'
3 import React from 'react';
4 import Link from 'next/link';
5
6 export default function CategoriesPage() {
7   // Categories ka list
8   const categories = ['Sport', 'Luxury', 'SUV', 'Sedan'];
9
10  return (
11    <div className="w-full flex flex-col items-center justify-center p-6">
12      <h1 className="text-2xl font-bold mb-6">Categories</h1>
13      <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4">
14        {categories.map((category, index) => (
15          <Link key={index} href={`/categories/${category.toLowerCase()}>
16            <div className="bg-[#eff2f4] p-6 rounded-lg text-center cursor-pointer">
17              <h2 className="text-lg font-semibold">{category}</h2>
18            </div>
19          </Link>
20        ))}
21    </div>
```

```
src > app > categories > [category] > page.tsx > ...
14 interface Car {
15   image?: {
16     _ref: string;
17     _type: "image";
18   };
19 };
20
21
22
23
24
25
26
27
28
29
30
31
32
33 export default function CategoryPage({ params }: { params: { category: string } }) {
34   const [cars, setCars] = useState<Car[]>([]);
35   const [loading, setLoading] = useState(true);
36   const [showMore, setShowMore] = useState(false);
37
38   // API se data fetch karna
39   useEffect(() => {
40     const fetchData = async () => {
41       try {
42         const response = await fetch('https://sanity-nextjs-application.vercel.app/api/hackathon3');
43         const data = await response.json();
44         setCars(data.cars);
45       } catch (error) {
46         console.error('Error fetching data:', error);
47       } finally {
48         setLoading(false);
49       }
50     };
51     fetchData();
52   });
53 }
```

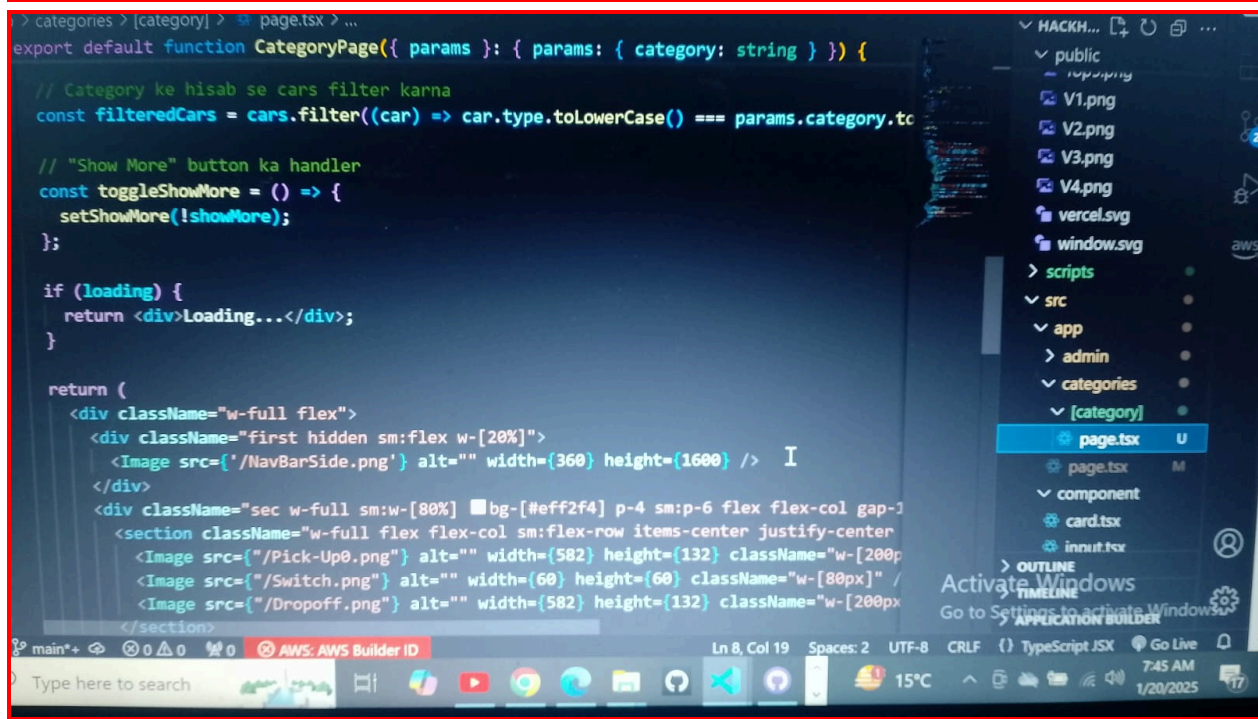
Dynamic /functionality Coding in multiples pages.



The screenshot shows the VS Code editor with the file `products.ts` open. The code defines an interface `Products` with the following properties:

```
export interface Products {
  _id :string;
  name :string;
  brand :string;
  type : string;
  fuelCapaci : string;
  transmission : string;
  seatingCapacity : string;
  pricePerDay: string;
  originalPrice :string;
  tags : string[];
  image?: {
    asset : {
      _ref : string;
      _type : "image";
    };
  };
};
```

The Explorer sidebar on the right shows the project structure with folders `src` and `lib`. The `src` folder contains files like `live.ts`, `queries.ts`, `schemaTypes`, `cars.ts`, `index.ts`, `env.ts`, and `structure.ts`. The `lib` folder contains `products.ts`, which is the current file being edited.



The screenshot shows the VS Code editor with the file `page.tsx` open. The code defines a function component `CategoryPage` that filters cars based on the category parameter. The component uses `useState` for `showMore` and `loading` states. The JSX returns a `div` with a flex layout, containing a `NavBarSide.png` image and a `section` with a background color and padding. The `section` contains three `Image` components: `Pick-Up0.png`, `Switch.png`, and `Dropoff.png`.

```
export default function CategoryPage({ params }: { params: { category: string } }) {
  // Category ke hisab se cars filter karna
  const filteredCars = cars.filter((car) => car.type.toLowerCase() === params.category.toLowerCase());

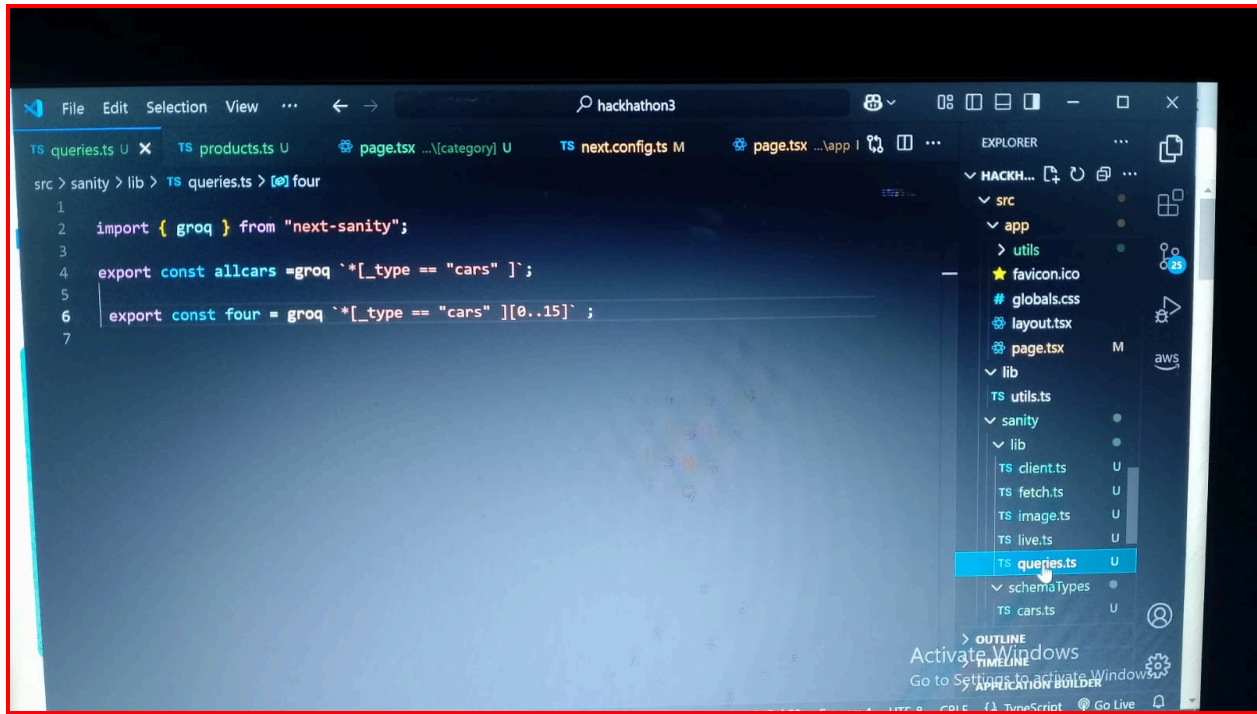
  // "Show More" button ka handler
  const toggleShowMore = () => {
    setShowMore(!showMore);
  };

  if (loading) {
    return <div>Loading...</div>;
  }

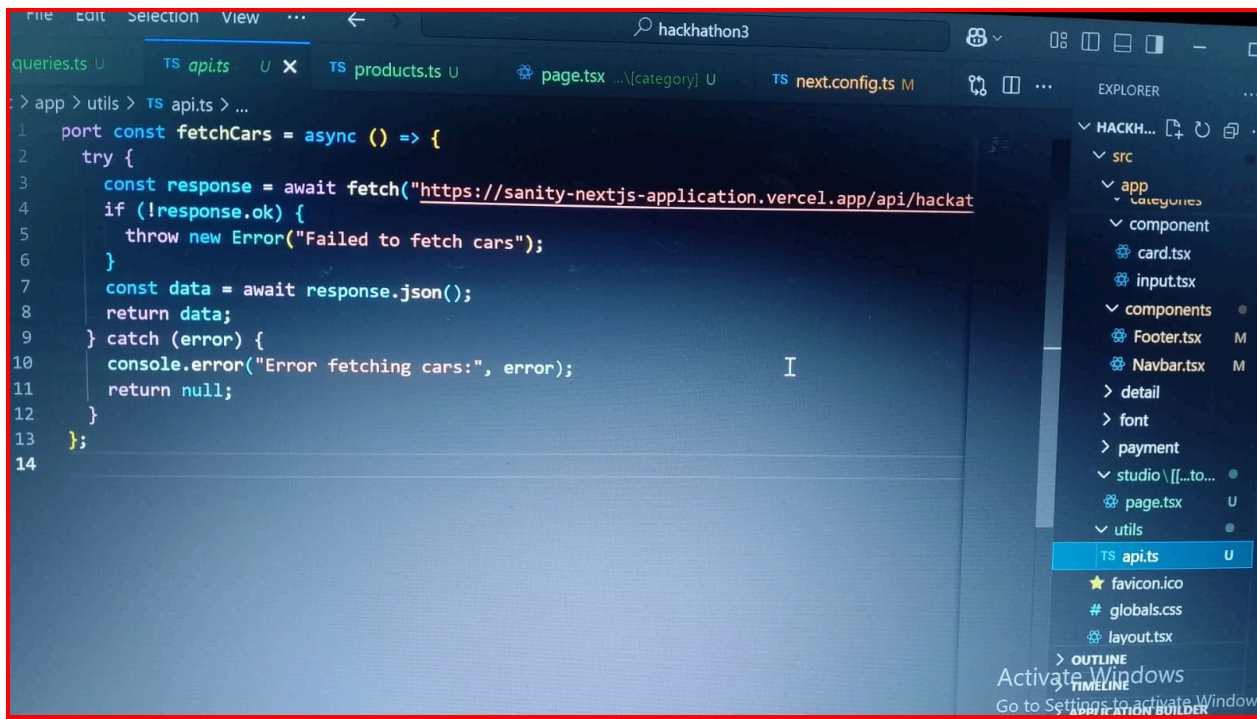
  return (
    <div className="w-full flex">
      <div className="first hidden sm:flex w-[20%]">
        <Image src="/NavBarSide.png" alt="" width={360} height={1600} />
      </div>
      <div className="sec w-full sm:w-[80%] bg-[#eff2f4] p-4 sm:p-6 flex flex-col gap-1">
        <section className="w-full flex flex-col sm:flex-row items-center justify-center">
          <Image src="/Pick-Up0.png" alt="" width={582} height={132} className="w-[200px] h-[100px]" />
          <Image src="/Switch.png" alt="" width={60} height={60} className="w-[80px] h-[80px]" />
          <Image src="/Dropoff.png" alt="" width={582} height={132} className="w-[200px] h-[100px]" />
        </section>
      </div>
    </div>
  );
}
```

The Explorer sidebar on the right shows the project structure with folders `public`, `scripts`, `src`, `app`, `admin`, `categories`, and `[category]`. The `[category]` folder contains `page.tsx`, which is the current file being edited.

Dynamic /functionality Coding in multiples pages.



```
src > sanity > lib > TS queries.ts > four
1
2 import { groq } from "next-sanity";
3
4 export const allcars = groq `*[_type == "cars" ]`;
5
6 export const four = groq `*[_type == "cars" ][0..15]`;
7
```



```
> app > utils > TS api.ts > ...
1 port const fetchCars = async () => {
2   try {
3     const response = await fetch("https://sanity-nextjs-application.vercel.app/api/hackat
4     if (!response.ok) {
5       throw new Error("Failed to fetch cars");
6     }
7     const data = await response.json();
8     return data;
9   } catch (error) {
10    console.error("Error fetching cars:", error);
11    return null;
12  }
13 };
14
```