

# Day 4 - Dynamic Frontend Components Report

## Project Overview

This project focuses on building a dynamic frontend for a car rental website using Next.js and Sanity CMS. The main goal is to implement functional, reusable components to enable users to filter cars based on attributes such as type, seating capacity, and fuel capacity.

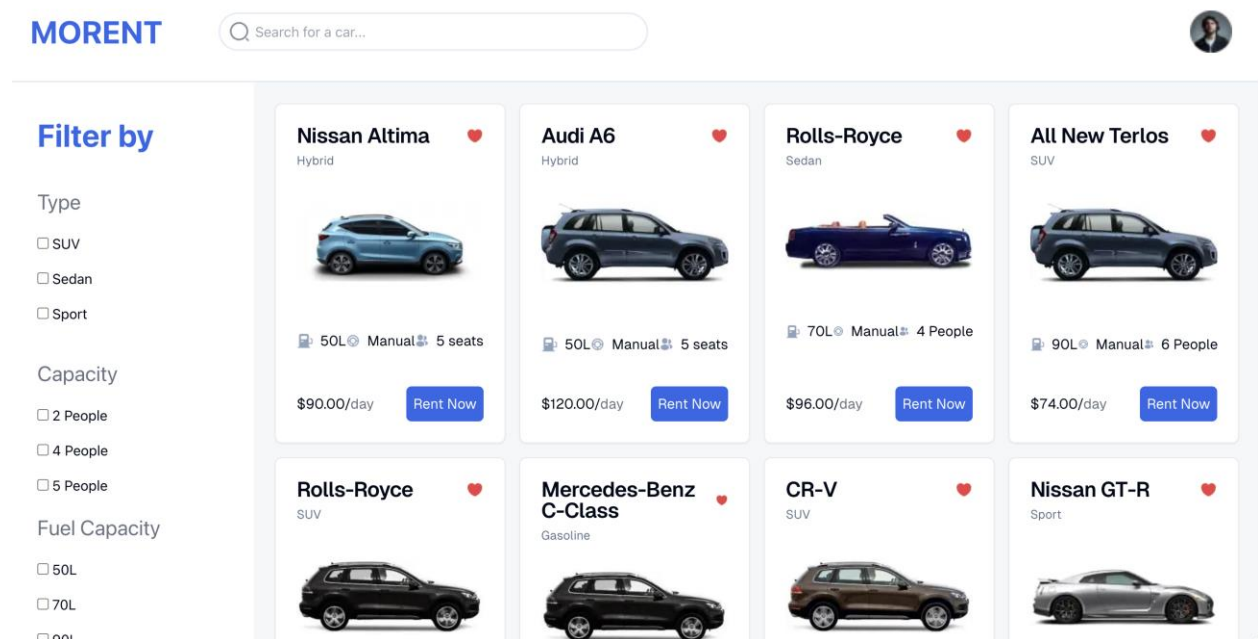
### Step 1: Planning the Filters

#### 1. Identified Attributes:

- Car type (SUV, Sedan, Sport)
- Seating capacity (2 People, 4 People, 5 People)
- Fuel capacity (50L, 70L, 90L)

#### 2. Defined Objectives:

- Enable users to filter cars dynamically.
- Ensure the filters are reusable and modular.



### Step 2: Setting Up the Filter Component

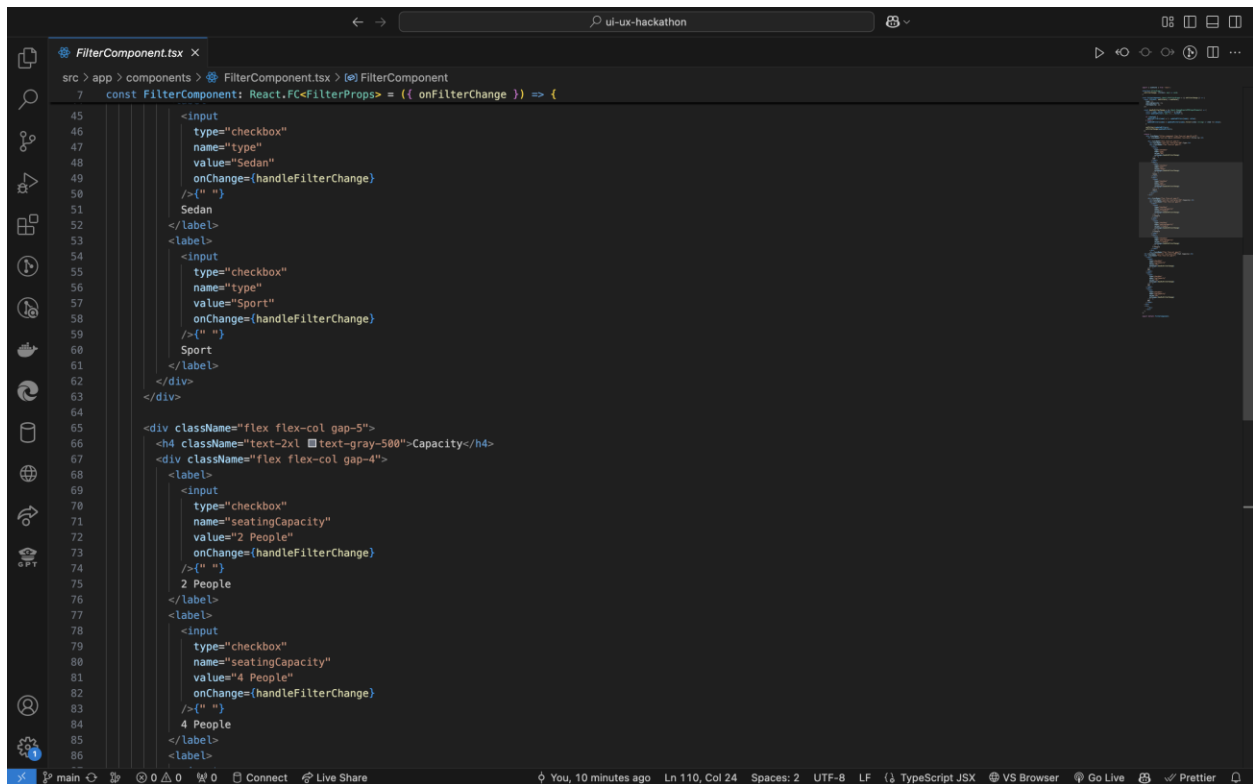
#### 1. Created a Reusable Component:

- a. Developed a FilterComponent using React functional components with TypeScript.
  - b. Integrated a useState hook to manage filter state locally.
- 2. Implemented Input Elements:**
  - a. Added checkbox inputs for each attribute.
  - b. Handled filter updates using an onChange event listener.
- 3. Code Implementation:**

```
src > app > components > FilterComponent.tsx > [0] FilterComponent
You, 10 minutes ago | 1 author (You)
1 import { useState } from 'react';
2
3 interface FilterProps {
4   onFilterChange: (filters: any) => void;
5 }
6
7 const FilterComponent: React.FC<FilterProps> = ({ onFilterChange }) => {
8   const [filters, setFilters] = useState({
9     type: [],
10    seatingCapacity: [],
11    fuelCapacity: [],
12  });
13
14  const handleFilterChange = (e: React.ChangeEvent<HTMLInputElement>) => {
15    const { name, value, checked } = e.target;
16    const updatedFilters: any = { ...filters };
17
18    if (checked) {
19      updatedFilters[name] = [...updatedFilters[name], value];
20    } else {
21      updatedFilters[name] = updatedFilters[name].filter((item: string) => item !== value);
22    }
23
24    setFilters(updatedFilters);
25    onFilterChange(updatedFilters);
26  };
27
28  return (
29    <div className="filter-component flex flex-col gap-10 p-10">
30      <h3 className="text-4xl text-[#3563e9] font-bold">Filter by</h3>
31
32      <div className="flex flex-col gap-5">
33        <h4 className="text-2xl text-gray-500">Type</h4>
34        <div className="flex flex-col gap-4">
35          <label>
36            <input
37              type="checkbox"
38              name="type"
39              value="SUV"
40              onChange={handleFilterChange}
41            />{" "}
42            SUV
```

interface HTMLInputElement  
Provides special properties and methods for manipulating the options, layout, and presentation of elements.  
[MDN Reference](#)

```
7 const FilterComponent: React.FC<FilterProps> = ({ onFilterChange }) => {
87
88   <input
89     type="checkbox"
90     name="seatingCapacity"
91     value="5 People"
92     onChange={handleFilterChange}
93   />{" "}
94   5 People
95 </div>
96 <div className="flex flex-col gap-5">
97 <h4 className="text-2xl text-gray-500">Fuel Capacity</h4>
98 <div className="flex flex-col gap-4">
99 <label>
100   <input
101     type="checkbox"
102     name="fuelCapacity"
103     value="50L"
104     onChange={handleFilterChange}
105   />{" "}
106   50L
107 </label>
108 <label>
109   <input
110     type="checkbox"
111     name="fuelCapacity"
112     value="70L"
113     onChange={handleFilterChange}
114   />{" "}
115   70L
116 </label>
117 <label>
118   <input
119     type="checkbox"
120     name="fuelCapacity"
121     value="60L"
122     onChange={handleFilterChange}
123   />{" "}
124   60L
125 </label>
126 </div>
127 </div>
128 </div>
```



```
src > app > components > FilterComponent.tsx > FilterComponent
7  const FilterComponent: React.FC<FilterProps> = ({ onFilterChange }) => {
45
46     <input
47       type="checkbox"
48       name="type"
49       value="Sedan"
50       onChange={handleFilterChange}
51     />{" "}
52     Sedan
53   </label>
54   <input
55     type="checkbox"
56     name="type"
57     value="Sport"
58     onChange={handleFilterChange}
59   />{" "}
60   Sport
61 </label>
62 </div>
63 </div>
64
65 <div className="flex flex-col gap-5">
66   <h4 className="text-2xl text-gray-500">Capacity</h4>
67   <div className="flex flex-col gap-4">
68     <label>
69       <input
70         type="checkbox"
71         name="seatingCapacity"
72         value="2 People"
73         onChange={handleFilterChange}
74       />{" "}
75       2 People
76     </label>
77     <label>
78       <input
79         type="checkbox"
80         name="seatingCapacity"
81         value="4 People"
82         onChange={handleFilterChange}
83       />{" "}
84       4 People
85     </label>
86   </div>
87 </div>
```

## Challenges Faced

### 1. Data Structure Complexity:

- Initially faced difficulty managing nested filter states.
- Resolved by creating a unified state object for all filters.

### 2. Performance Optimization:

- Filtering large datasets caused noticeable lag.
- Improved efficiency by applying filters sequentially in the applyFilters function.

## Outcomes and Learnings

### 1. Functional Filters:

- Successfully implemented dynamic filtering for car type, seating capacity, and fuel capacity.

### 2. Improved React Skills:

- Gained deeper understanding of state management and reusable component design.

### **3. Sanity Integration:**

- a. Enhanced familiarity with GROQ queries and Sanity's CMS capabilities.

## **Code Deliverables**

- Search bar

```
src > app > components > Search.tsx > Search > handleSearch > filtered > data.filter() callback
You, 4 hours ago | 1 author (You)
1 'use client';
2
3 import React, { useState } from 'react';
4 import Image from 'next/image';
5 import Link from 'next/link';
6 import { Card, CardContent, CardDescription, CardFooter, CardHeader, CardTitle } from '@components/ui/card';
7 import { urlFor } from '../lib/sanity';
8
9 You, 6 hours ago | 1 author (You)
10 interface SimplifiedCar {
11   _id: string;
12   name: string;
13   type: string;
14   slug: { current: string };
15   image: string;
16   fuelCapacity: string;
17   transmission: string;
18   seatingCapacity: string;
19   pricePerDay: string;
20 }
21
22 You, 6 hours ago | 1 author (You)
23 interface SearchProps {
24   data: SimplifiedCar[];
25 }
26
27 export default function Search({ data }: SearchProps) {
28   const [searchQuery, setSearchQuery] = useState('');
29   const [filteredCars, setFilteredCars] = useState<SimplifiedCar[]>([]);
30
31   const handleSearch = (e: React.ChangeEvent<HTMLInputElement>) => {
32     const query = e.target.value;
33     setSearchQuery(query);
34
35     if (query.trim()) {
36       const filtered = data.filter((car) => {
37         console.log(car.name.toLowerCase());
38         return car.name.toLowerCase().includes(query.toLowerCase());
39       });
40       setFilteredCars(filtered);
41     } else {
42       setFilteredCars([]);
43     }
44     setSearchQuery('');
45   };
46 }
```

```
Search.tsx
src > app > components > Search.tsx > Search > handleSearch > filtered > data.filter() callback
25 export default function Search({ data }: SearchProps) {
45   return (
46     <div className="relative w-full md:w-[492px]">
47       <Image
48         src="/search-normal.png"
49         alt="Search"
50         width={24}
51         height={24}
52         className="absolute top-1/2 left-3 transform -translate-y-1/2"
53       />
54       <input
55         type="text"
56         value={searchQuery}
57         onChange={handleSearch}
58         placeholder="Search for a car..."
59         className="border-2 border-[#e7eef6] w-full h-[44px] rounded-full p-2 pl-10 pr-12 focus:outline-none focus:border-[#3563e9]"
60       />
61       {searchQuery && filteredCars.length > 0 && (
62         <div className="absolute z-50 w-full mt-2 bg-white rounded-lg shadow-lg border border-[#e7eef6] max-h-[480px] overflow-y-auto">
63           <div className="p-4">
64             {filteredCars.map((car) => (
65               <Card
66                 key={car._id}
67                 className="mb-4 shadow-lg border rounded-md p-2 hover:bg-gray-100"
68               >
69                 <CardHeader>
70                   <CardTitle className="flex justify-between items-center">
71                     {car.name}
72                     <Image
73                       src="/heart.png"
74                       alt="Favorite"
75                       width={20}
76                       height={20}
77                     />
78                   </CardTitle>
79                   <CardDescription>{car.type}</CardDescription>
80                 </CardHeader>
81                 <CardContent className="flex flex-col items-center gap-4">
82                   <Image
83                     src={urlFor(car.image).url()}
84                     alt={car.name}
85                     width={220}
86                     height={68}
87                   />
88                 </CardContent>
89               </Card>
90             )
91           )}
92         </div>
93       )}
94     </div>
95   );
96 }
```

```
src > app > components > Search.tsx > Search > handleSearch > filtered > data.filter() callback
25 export default function Search({ data }: SearchProps) {
64   {filteredCars.map(car) => (
98     <div className="flex items-center gap-2">
99       <Image
100         src="/Caricon.png"
101         alt="Transmission"
102         width={26}
103         height={24}
104       />
105       <span>{car.transmission}</span>
106     </div>
107     <div className="flex items-center gap-2">
108       <Image
109         src="/profile-2user.png"
110         alt="Seating Capacity"
111         width={26}
112         height={24}
113       />
114       <span>{car.seatingCapacity}</span>
115     </div>
116   </CardContent>
117   <CardFooter className="flex justify-between items-center">
118     <p>
119       {car.pricePerDay}/<span className="text-gray-500">day</span>
120     </p>
121     <Link href={`/categories/${car.slug.current}`}>
122       <button className="bg-[#3563e9] p-2 text-white rounded-md" onClick={() => setSearchQuery('')}>
123         Rent Now
124       </button>
125     </Link>
126   </CardFooter>
127 </Card>
128   ))
129 </div>
130 </div>
131 </div>
132 </div>
133 {searchQuery && filteredCars.length === 0 && (
134   <div className="absolute z-50 w-full mt-2 bg-white rounded-lg shadow-lg border border-[#eef6] p-4 text-center">
135     No cars found matching "{searchQuery}"
136   </div>
137 )}
138 </div>
139 </div>
);
```

## Conclusion

The "Dynamic Frontend Components" project successfully demonstrates the creation of a user-friendly, filterable and searchable car rental interface. The modular approach ensures scalability and ease of future development.