Day 4 - Dynamic Frontend Components

1. Functional Deliverables

Screenshots or Screen Recordings

Attach the following visuals:

- Product Listing Page with dynamic data.
- Individual Product Detail Pages with accurate routing and data rendering.
- Working Category Filters, Search Bar, and Pagination.
- Any additional features, such as related products or user profile components.

---

2. Code Deliverables

Key Components Code Snippets

ProductCard Component

```
import React from 'react';

const ProductCard = ({ product }) => {
  return (
    <div className="border rounded-lg p-4 shadow-md">
      <img src={product.image} alt={product.name} className="w-full h-40 object-cover"
      <h3 className="text-lg font-bold mt-2">{product.name}</h3>
      <p className="text-gray-500">{product.brand}</p>
      <p className="text-blue-500 font-semibold">${product.pricePerDay}/day</p>
    </div>
  );
};

export default ProductCard;
```

ProductList Component

```
import React, { useState, useEffect } from 'react';
import ProductCard from './ProductCard';

const ProductList = () => {
  const [products, setProducts] = useState([]);

  useEffect(() => {
    fetch('/api/products')
      .then(response => response.json())
      .then(data => setProducts(data));
  }, []);

  return (
    <div className="grid grid-cols-3 gap-4">
      {products.map((product) => (
        <ProductCard key={product.id} product={product} />
```

SearchBar Component

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

const SearchBar = ({ onSearch }) => {
  const [query, setQuery] = useState('');

  const handleSearch = (e) => {
    setQuery(e.target.value);
    onSearch(e.target.value);
  };

  return (
    <input
      type="text"
      placeholder="Search products..."
      value={query}
      onChange={handleSearch}
      className="border rounded-lg p-2 w-full"
    />
  );
};

export default SearchBar;
```

API Integration and Dynamic Routing

- Fetching products via API.
- Dynamic routing for product details using Next.js routes.
- Implementing category filters and pagination.

---

3. Documentation

Technical Report

Steps Taken

1. Set up Next.js project with Tailwind CSS.
2. Created ProductCard, ProductList, and SearchBar components.
3. Integrated API to fetch and display product data dynamically.
4. Implemented category filters, pagination, and search functionality.
5. Added dynamic routing for individual product pages.

Challenges & Solutions

- **Handling API errors:** Implemented try-catch blocks and loading states.
- **Optimizing performance:** Used React.memo and lazy loading for images.

Best Practices Followed

- Component-based architecture.
- Code reusability and modularity.
- Proper error handling and API optimization.