DAY 4: **TECHNICAL REPORT ON COMPONENT DEVELOPMENT AND INTEGRATION**

1. Steps Taken to Build and Integrate Components

Requirement Analysis:

- Identified the purpose and functionality of each component (e.g., ProductCard for displaying product details).
- Defined key props for components like title, image, price, and description.

Component Creation:

- Built reusable functional components using TypeScript and React.
- Ensured proper typing of props using TypeScript interfaces or types.
- Example:
- type ProductCardProps = {
-   title: string;
-   image: string;
-   price: number;
- };
-
- const ProductCard: React.FC<ProductCardProps> = ({ title, image, price }) => (
-   <div>
-     <h2>{title}</h2>

- `    <img src={image} alt={title} />`
- `    <p>{price}</p>`
- `  </div>`
- `);`

Styling Integration:

- Used **CSS Modules** or **Tailwind CSS** for styling to maintain modularity and avoid global conflicts.
- Organized styles in dedicated files for each component (e.g., ProductCard.module.css).

Dynamic Data Integration:

- Connected components to fetch or render data from APIs or state management solutions (like Redux or Context API).
- Example:
- const products = useSelector((state) => state.products);

Testing and Debugging:

- Utilized unit testing frameworks like Jest or React Testing Library to ensure components function correctly.
- Validated props and handled edge cases.

Integration:

- Combined individual components into pages (e.g., integrating ProductCard into a ProductList page).
- Used routing (Next.js or React Router) to navigate between different views.

---

2. Challenges Faced and Solutions Implemented

Challenge: TypeScript Errors with Props

- Errors like TS2607 (JSX element class does not support attributes).
- **Solution:** Properly defined props types/interfaces and ensured components accepted props.

Challenge: Unrecognized JSX Components

- Components like Image were not recognized.

- **Solution:** Imported the correct Image component from libraries (e.g., next/image) or replaced it with native img.

Challenge: CSS Conflicts

- Styling issues due to conflicting global styles.
- **Solution:** Adopted CSS Modules or Tailwind CSS for scoped and utility-based styling.

Challenge: Inline Style Warning

- Received warnings about inline styles.
- **Solution:** Moved styles to external CSS files or CSS Modules.

Challenge: Component Reusability

- Some components were too specific and not reusable.
- **Solution:** Refactored components to be more generic by passing props for customizable behavior.

---

3. Best Practices Followed During Development

Modular Design:

- Built small, focused, and reusable components to reduce redundancy and improve maintainability.

TypeScript for Type Safety:

- Used TypeScript to define props and state types, reducing runtime errors and improving code clarity.

Version Control:

- Managed code using Git, committing changes frequently with descriptive commit messages.

Error Handling:

- Added error boundaries and validation to handle unexpected data or API failures.

Responsive Design:

- Ensured components were fully responsive by using CSS Flexbox/Grid and testing across multiple devices.

Code Review and Collaboration:

- Conducted peer reviews to ensure code quality and alignment with project goals.

Performance Optimization:

- Used lazy loading for images and code-splitting techniques to enhance page performance.

Conclusion

The development process followed a structured approach, tackling challenges effectively and adhering to best practices. These steps ensured the creation of robust, scalable, and maintainable components that meet project requirements. Continuous testing and debugging improved the reliability and usability of the integrated system.

Screen Recording

Screen recording is in zip folder.

. Code Deliverables:

Cart Button:

```
'use client';

import { useDispatch } from 'react-redux';
import { addToCart } from '@/app/store/CartSlice';
import { Button } from './ui/button';
import { Image as IImage } from 'sanity';
import { useRouter } from "next/navigation";
import { FC } from 'react';

// Define Props Type
interface AddToCartButtonProps {
  product: {
    _id: string;
    title: string;
    price: number;
    image: IImage;
    description: string;
  };
  children: React.ReactNode;
}

const AddToCartButton: FC<AddToCartButtonProps> = ({ children, product }) => {
  const dispatch = useDispatch();
  const router = useRouter();

  const handleAddToCart = () => {
    const cartItem = { ...product, quantity: 1 }; // Add default quantity
    dispatch(addToCart(cartItem));
    console.log('Added to Cart:', cartItem); // Debug log
    router.push("/Cart");
  };

  return (
    <Button
      className="bg-[rgba(2,159,174,1)] relative  p-4 rounded"
      onClick={handleAddToCart}
    >
      {children}
    </Button>
  );
};

export default AddToCartButton;
```

Product Card:

```
'use client';

import React, { useEffect, useState } from "react";
import { Image as IImage } from 'sanity';
import { Button } from "@/components/ui/button";
import { urlFor } from '@/sanity/lib/image';
import Link from 'next/link';
import AddToCartButton from "@/components/AddToCartButton";
import { client } from "@/lib/SanityClient";
import Image from "next/image";

export const getProductData = async () => {
  return client.fetch(`*[_type == "product"] {
    title,
    price,
    image,
    "slug": slug.current,
    _id,
    quantity,
    description
  }`);
};

export interface IProduct {
  title: string;
  price: number;
```

```
 image: IImage;
 _id: string;
 slug: string;
 quantity: number;
 description: string;
}

const ProductCard = () => {
 const [data, setData] = useState<IProduct[]>([]);
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  const fetchData = async () => {
   try {
    const res = await getProductData();
    setData(res);
   } catch (error) {
    console.error("Error fetching product data:", error);
   } finally {
    setLoading(false);
   }
  };
  fetchData();
 }, []);

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

 if (data.length === 0) {
  return <div>No products available.</div>;
 }

 return (
  <>
   {data.map((item) => (
    <div key={item._id} className="bg-white rounded-lg shadow-md overflow-hidden">
     <div
      className="h-64 bg-cover bg-center transition-transform duration-300 hover:scale-105 relative"
      style={{ backgroundImage: `url(${urlFor(item.image).url()})` }}
      aria-label={item.title}
     >
      <Button variant="secondary" size="lg" className="mt-3 ml-3 text-white">
       New
      </Button>
     </div>
     <div className="p-4 flex justify-between">
```

```
          <div>
            <Link
              href={`/AllProducts/${item.slug}`}
              className="text-lg font-normal text-[rgba(0,117,128,1)] hover:underline"
            >
              {item.title}
            </Link>
            <div className="text-xl mt-2 text-[rgba(39,35,67,1)]">${item.price}</div>
          </div>
          <div>
            <AddToCartButton product={item}>
              <Image src="/Group.png" alt="Add to Cart" width={18.5} height={18.4} />
            </AddToCartButton>
          </div>
        </div>
      </div>
    ))}
  </>
 );
};

export default ProductCard;
```