Day 4 - Dynamic Frontend Components - ShopVista

1. Functional Deliverables

Product Listing Page

- Implemented a dynamic product listing page that fetches and displays product data from the API.

- Each product is displayed using a reusable `ProductCard` component.

- Includes category filters, a search bar, and pagination for better user navigation.

Individual Product Detail Pages

- Implemented dynamic routing to generate individual product detail pages.

- Each page accurately renders product details including images, price, description, and related products.

- Optimized API calls to fetch data efficiently.

Category Filters, Search Bar, and Pagination

- Category Filters: Allows users to filter products based on categories.

- Search Bar: Enables users to search for products dynamically.

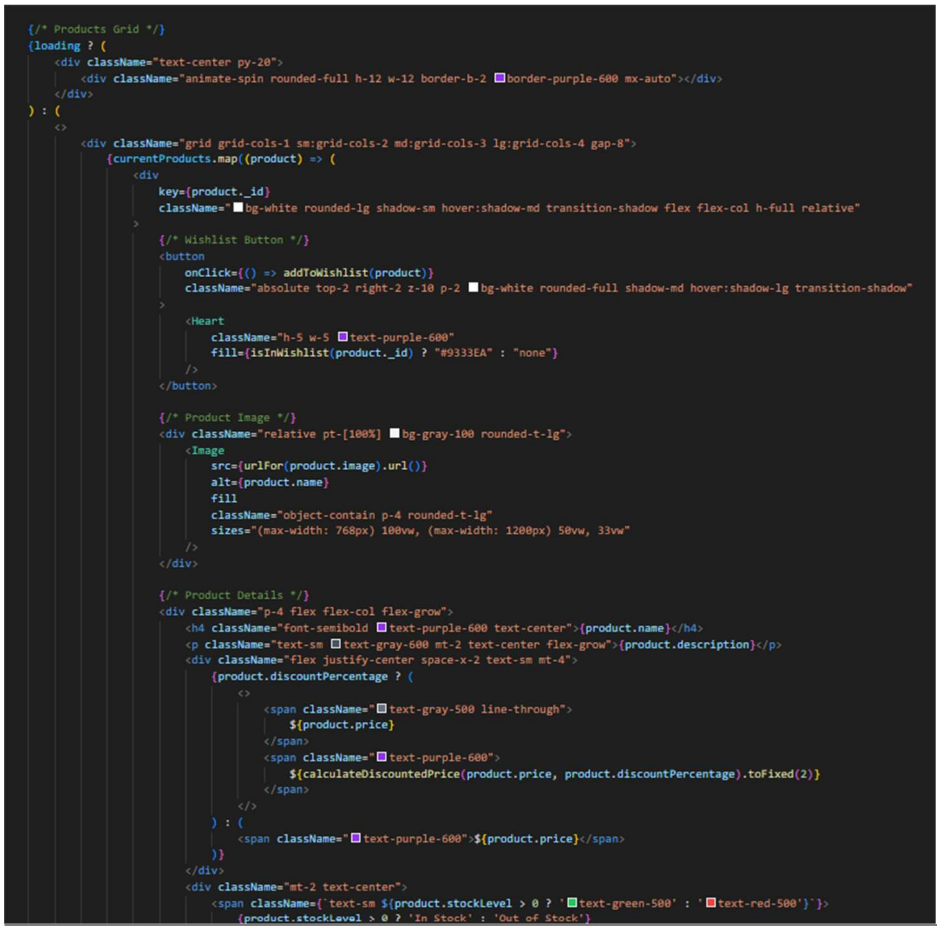
- Pagination: Implemented to handle large datasets efficiently.

Video Link :

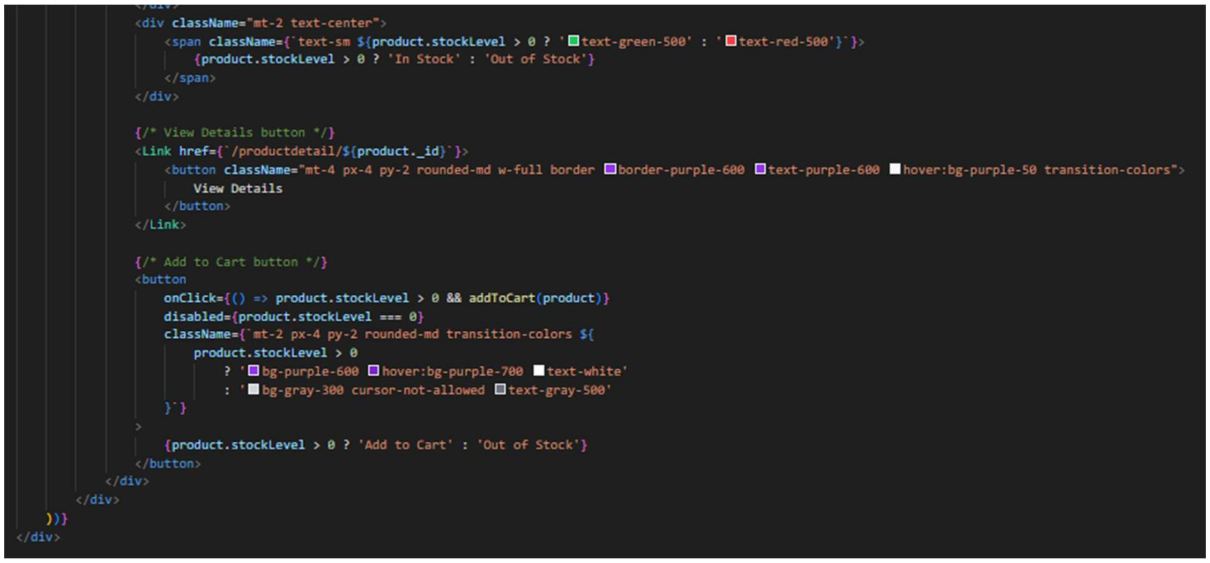
Day4_Dynamic_Frontend.mp4

2. Code Deliverables:

Product List:

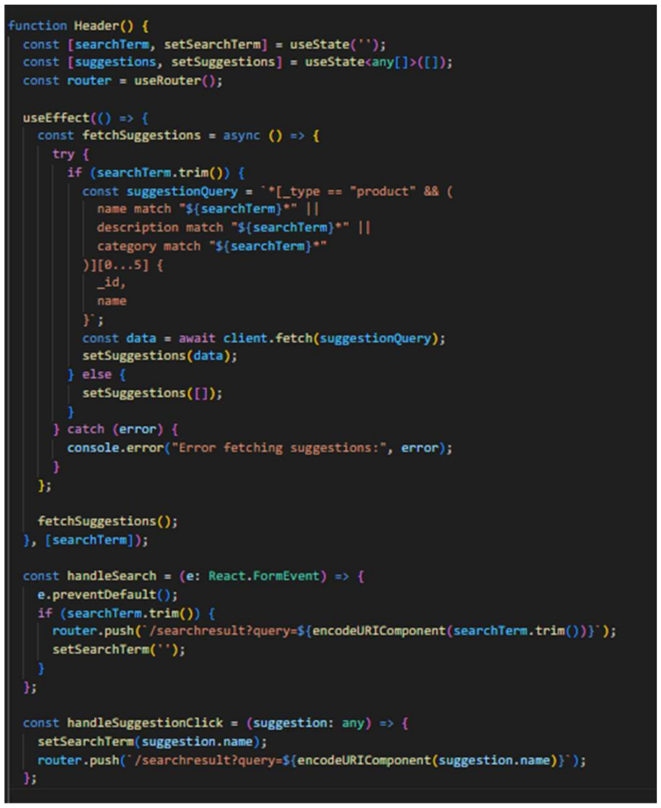


```
{/* Products Grid */}
{loading ? (
    <div className="text-center py-20">
        <div className="animate-spin rounded-full h-12 w-12 border-b-2 border-purple-600 mx-auto"></div>
    </div>
) : (
    <>
        <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-8">
            {currentProducts.map((product) => (
                <div
                    key={product._id}
                    className="bg-white rounded-lg shadow-sm hover:shadow-md transition-shadow flex flex-col h-full relative"
                >
                    {/* Wishlist Button */}
                    <button
                        onClick={() => addToWishlist(product)}
                        className="absolute top-2 right-2 z-10 p-2 bg-white rounded-full shadow-md hover:shadow-lg transition-shadow"
                    >
                        <Heart
                            className="h-5 w-5 text-purple-600"
                            fill={isInWishlist(product._id) ? "#9333EA" : "none"}
                        />
                    </button>

                    {/* Product Image */}
                    <div className="relative pt-[100%] bg-gray-100 rounded-t-lg">
                        <Image
                            src={urlFor(product.image).url()}
                            alt={product.name}
                            fill
                            className="object-contain p-4 rounded-t-lg"
                            sizes="(max-width: 768px) 100vw, (max-width: 1200px) 50vw, 33vw"
                        />
                    </div>

                    {/* Product Details */}
                    <div className="p-4 flex flex-col flex-grow">
                        <h4 className="font-semibold text-purple-600 text-center">{product.name}</h4>
                        <p className="text-sm text-gray-600 mt-2 text-center flex-grow">{product.description}</p>
                        <div className="flex justify-center space-x-2 text-sm mt-4">
                            {product.discountPercentage ? (
                                <>
                                    <span className="text-gray-500 line-through">
                                        ${product.price}
                                    </span>
                                    <span className="text-purple-600">
                                        ${calculateDiscountedPrice(product.price, product.discountPercentage).toFixed(2)}
                                    </span>
                                </>
                            ) : (
                                <span className="text-purple-600">${product.price}</span>
                            )}
                        </div>
                        <div className="mt-2 text-center">
                            <span className={`text-sm ${product.stockLevel > 0 ? 'text-green-500' : 'text-red-500'}`}>
                                {product.stockLevel > 0 ? 'In Stock' : 'Out of Stock'}
```



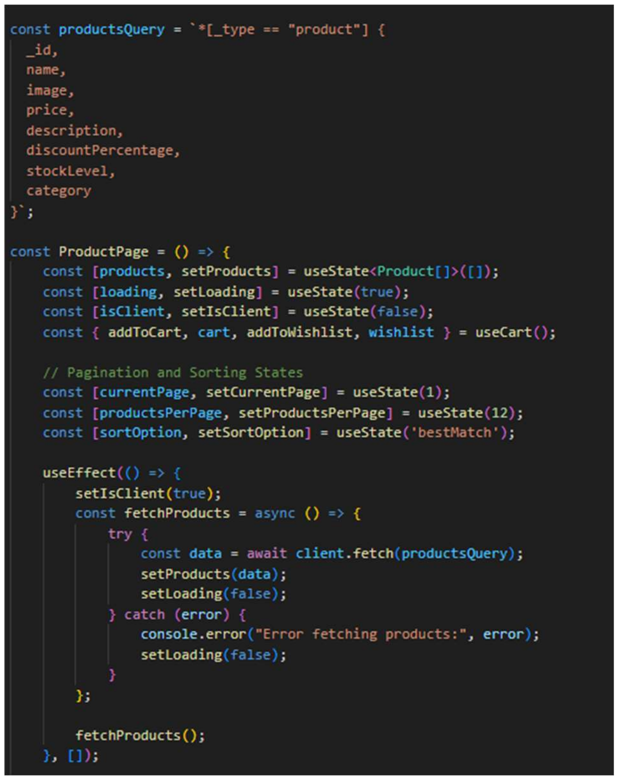
```
                        <div className="mt-2 text-center">
                            <span className={`text-sm ${product.stockLevel > 0 ? 'text-green-500' : 'text-red-500'}`}>
                                {product.stockLevel > 0 ? 'In Stock' : 'Out of Stock'}
                            </span>
                        </div>

                        {/* View Details button */}
                        <Link href={`/productdetail/${product._id}`}>
                            <button className="mt-4 px-4 py-2 rounded-md w-full border border-purple-600 text-purple-600 hover:bg-purple-50 transition-colors">
                                View Details
                            </button>
                        </Link>

                        {/* Add to Cart button */}
                        <button
                            onClick={() => product.stockLevel > 0 && addToCart(product)}
                            disabled={product.stockLevel === 0}
                            className={`mt-2 px-4 py-2 rounded-md transition-colors ${
                                product.stockLevel > 0
                                    ? 'bg-purple-600 hover:bg-purple-700 text-white'
                                    : 'bg-gray-300 cursor-not-allowed text-gray-500'
                            }`}
                        >
                            {product.stockLevel > 0 ? 'Add to Cart' : 'Out of Stock'}
                        </button>
                    </div>
                </div>
            ))}
        </div>
```

Search Bar:



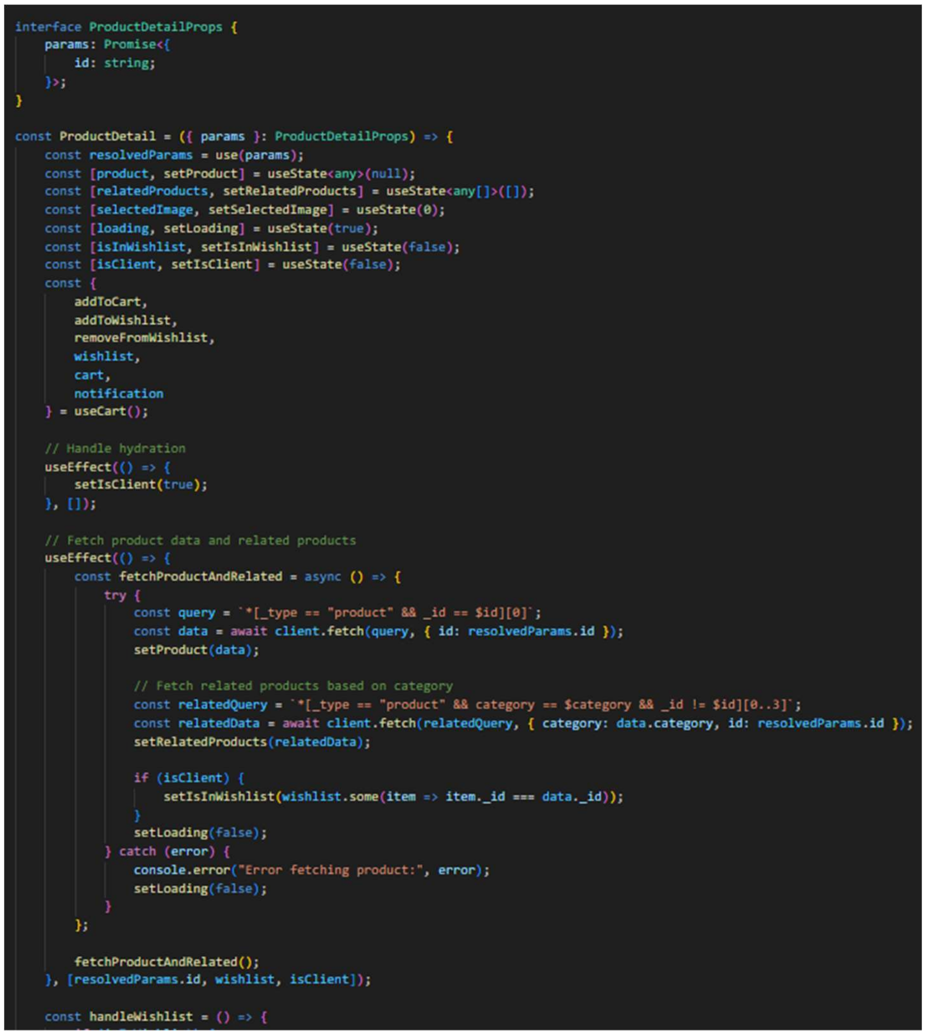
```
function Header() {
  const [searchTerm, setSearchTerm] = useState('');
  const [suggestions, setSuggestions] = useState<any[]>([]);
  const router = useRouter();

  useEffect(() => {
    const fetchSuggestions = async () => {
      try {
        if (searchTerm.trim()) {
          const suggestionQuery = `*[_type == "product" && (
            name match "${searchTerm}*" ||
            description match "${searchTerm}*" ||
            category match "${searchTerm}*"
          )][0...5] {
            _id,
            name
          }`;
          const data = await client.fetch(suggestionQuery);
          setSuggestions(data);
        } else {
          setSuggestions([]);
        }
      } catch (error) {
        console.error("Error fetching suggestions:", error);
      }
    };

    fetchSuggestions();
  }, [searchTerm]);

  const handleSearch = (e: React.FormEvent) => {
    e.preventDefault();
    if (searchTerm.trim()) {
      router.push(`/searchresult?query=${encodeURIComponent(searchTerm.trim())}`);
      setSearchTerm('');
    }
  };

  const handleSuggestionClick = (suggestion: any) => {
    setSearchTerm(suggestion.name);
    router.push(`/searchresult?query=${encodeURIComponent(suggestion.name)}`);
  };
```

Api Integration:

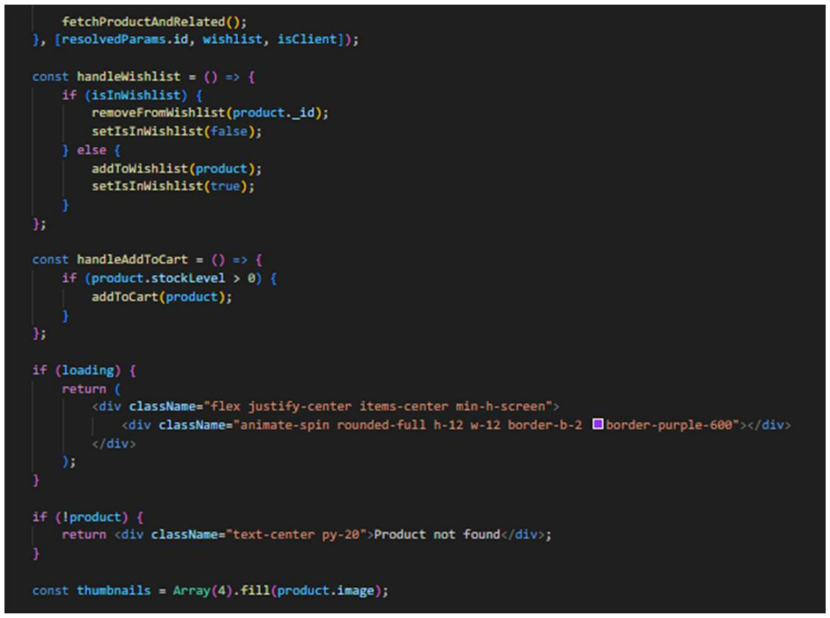


```
const productsQuery = `*[_type == "product"] {
  _id,
  name,
  image,
  price,
  description,
  discountPercentage,
  stockLevel,
  category
}`;

const ProductPage = () => {
    const [products, setProducts] = useState<Product[]>([]);
    const [loading, setLoading] = useState(true);
    const [isClient, setIsClient] = useState(false);
    const { addToCart, cart, addToWishlist, wishlist } = useCart();

    // Pagination and Sorting States
    const [currentPage, setCurrentPage] = useState(1);
    const [productsPerPage, setProductsPerPage] = useState(12);
    const [sortOption, setSortOption] = useState('bestMatch');

    useEffect(() => {
        setIsClient(true);
        const fetchProducts = async () => {
            try {
                const data = await client.fetch(productsQuery);
                setProducts(data);
                setLoading(false);
            } catch (error) {
                console.error("Error fetching products:", error);
                setLoading(false);
            }
        };

        fetchProducts();
    }, []);
```

Product Card:



```
interface ProductDetailProps {
    params: Promise<{
        id: string;
    }>;
}

const ProductDetail = ({ params }: ProductDetailProps) => {
    const resolvedParams = use(params);
    const [product, setProduct] = useState<any>(null);
    const [relatedProducts, setRelatedProducts] = useState<any[]>([]);
    const [selectedImage, setSelectedImage] = useState(0);
    const [loading, setLoading] = useState(true);
    const [isInWishlist, setIsInWishlist] = useState(false);
    const [isClient, setIsClient] = useState(false);
    const {
        addToCart,
        addToWishlist,
        removeFromWishlist,
        wishlist,
        cart,
        notification
    } = useCart();

    // Handle hydration
    useEffect(() => {
        setIsClient(true);
    }, []);

    // Fetch product data and related products
    useEffect(() => {
        const fetchProductAndRelated = async () => {
            try {
                const query = `*[_type == "product" && _id == $id][0]`;
                const data = await client.fetch(query, { id: resolvedParams.id });
                setProduct(data);

                // Fetch related products based on category
                const relatedQuery = `*[_type == "product" && category == $category && _id != $id][0..3]`;
                const relatedData = await client.fetch(relatedQuery, { category: data.category, id: resolvedParams.id });
                setRelatedProducts(relatedData);

                if (isClient) {
                    setIsInWishlist(wishlist.some(item => item._id === data._id));
                }
                setLoading(false);
            } catch (error) {
                console.error("Error fetching product:", error);
                setLoading(false);
            }
        };

        fetchProductAndRelated();
    }, [resolvedParams.id, wishlist, isClient]);

    const handleWishlist = () => {
```



```
        fetchProductAndRelated();
    }, [resolvedParams.id, wishlist, isClient]);

    const handleWishlist = () => {
        if (isInWishlist) {
            removeFromWishlist(product._id);
            setIsInWishlist(false);
        } else {
            addToWishlist(product);
            setIsInWishlist(true);
        }
    };

    const handleAddToCart = () => {
        if (product.stockLevel > 0) {
            addToCart(product);
        }
    };

    if (loading) {
        return (
            <div className="flex justify-center items-center min-h-screen">
                <div className="animate-spin rounded-full h-12 w-12 border-b-2 ▪border-purple-600"></div>
            </div>
        );
    }

    if (!product) {
        return <div className="text-center py-20">Product not found</div>;
    }

    const thumbnails = Array(4).fill(product.image);
```

Best Practices Followed

- Used reusable components for better code maintainability.

- Optimized API calls to minimize load time.

- Followed proper file structuring for scalability.

- Implemented responsive UI for better accessibility.

GitHub Repository Link:

https://github.com/MehakFaheem/Hackathon-3.git