**Report:ImplementingaDynamicProductListingComponent**

**PreparedBy:**Bilawal Akber
**PreparedFor:**Project Day4 –Building Dynamic Frontend Components

**Objective:**
TheprimaryobjectiveofDay4istodesignanddevelop**dynamicfrontendcomponents** thatcan display marketplace data fetched from **Sanity CMS** or external APIs. This process focuses on modularity, reusability, and applying real-world development practices to build scalable and responsive web applications.

---

## TaskOverview

### Objective:

Builda**ProductListingComponent** foramarketplace.

### Requirements:

1. FetchproductdatadynamicallyusingSanityCMSoranexternalAPI.
2. Displaythedatain a**grid layout**of cardswith thefollowing details:
    - **ProductName**
    - **Price**
    - **Image**
    - **Stock Status**
3. Ensureresponsivenessacrossdevices.
4. Implementmodularitybybreakingthecomponentintosmaller,reusableparts.

### Tools &Technologies:

- **Framework:**ReactorNext.js
- **CMS:**SanityCMS
- **Styling:**TailwindCSSorplainCSS
- **StateManagement:**ReactHooks

---

## ImplementationPlan

1. **SetUpDataFetching:**
   - IntegrateSanityCMSorAPIendpointstofetchtheproductdatadynamically.
   - UseReacthooks(`useEffect`)fordatafetchingand(`useState`)tostoreand manage the data.
2. **DesignReusableComponents:**
   - BreakdowntheProduct ListingComponentintosmaller parts:
     - **ProductCardComponent:**Displaysindividualproduct details.
     - **GridLayoutComponent:**Arrangestheproductcardsinaresponsive grid.
3. **Apply Responsive Design:**
   - UseTailwindCSSorCSSGrid/Flexboxtoensurethegridlayoutadaptstoall screen sizes.
4. **EnhanceUser Experience:**
   - Highlightimportant detailslikestock statuswith conditional formatting.
   - Addhovereffects forbetter interactivity.

```
useEffect(() => {
    const fetchProducts = async () => {
      const productsData = await client.fetch(
        `*[_type == "food"]{
          name,
          price,
          description,
          category,
          originalPrice,
          "image": image.asset->url,
          "slug": slug.current,
        }`
      );
      setProducts(productsData);
      setFilteredProducts(productsData);
    };
    fetchProducts();
  }, []);
```

## 2. ProductDetailComponent

### Objective:

Developindividualproductdetailpagesusing**dynamicroutinginNext.js**.Thesepageswill display detailed information about each product, including:

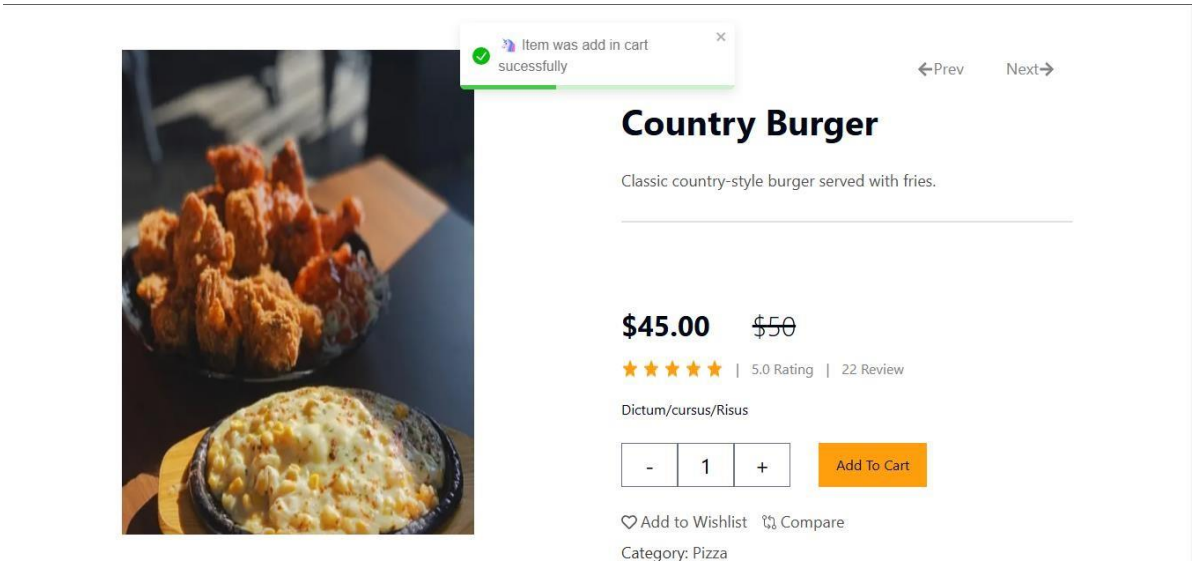
- **Name**
- **ProductDescription**
- **Price**
- **Category**
- **StockAvailability**

### ImplementationPlan:

1. **DynamicRouting:**
   - Createdynamicroutesusingthe `[id].tsx`fileinthe`pages/products`directory.
   - FetchproductdatabasedontheproductIDfromaCMSlikeSanityoranAPI.
2. **Data Fields:**
   Eachproductdetailpage shouldincludethefollowing fields:
   - **ProductDescription:**Adetailedexplanationoftheproduct,fetchedfromthe backend.
   - **Price:**Displayedprominentlyforclearvisibility.
3. **IntegrationwithProductListing:**
   - Linkeachproductcardinthe **ProductListingComponent**toitscorrespondingdetail page using the `Link`component in Next.js.
4. **StylingandLayout:**
   - UseTailwindCSSorplainCSSforacleanandresponsive design.
   - Ensurethelayouthighlightstheproductdescriptionandpriceforuserclarity.

```
1  async function Productpage({ params }: { params: { slug: string } }) {
2    const product:IProduct =
3      await client.fetch(`*[_type == "food" && slug.current == $slug][0] {
4        name,
5        description,
6        price,
7        originalPrice,
8        tags,
9        "imageUrl": image.asset->url,
10       "slug": slug.current,
11     }`,{slug:params.slug});
```

***UIDisplayOFProductDetailPage:***



## Step3:SearchBarwithPriceFilter

### Objective:

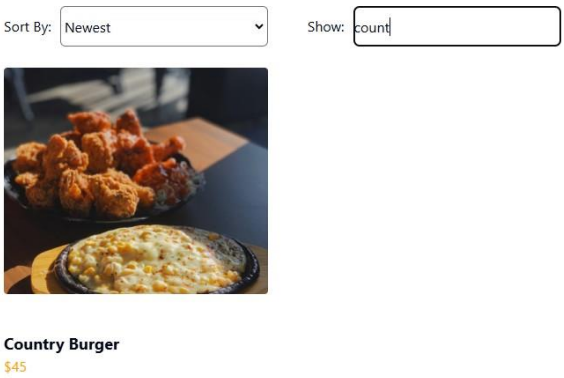
Toimplementa **searchbar** and**price filters**to enhancethe productbrowsing experience.

### ImplementationPlan:

1. **SearchBar Functionality:**
   - Filterproductsbasedontheirnameorassociatedtags.
   - Updatetheproductlistinreal-timeastheuser types.

```
// Handle search
 const handleSearch = (event: React.ChangeEvent<HTMLInputElement>) => {
   const query = event.target.value.toLowerCase();
   setSearchQuery(query);

   const filtered = products.filter(
     (product) =>
       product.name.toLowerCase().includes(query) ||
       product.description.toLowerCase().includes(query) ||
       product.category.toLowerCase().includes(query) ||
       product.slug.toLowerCase().includes(query)
   );
   setFilteredProducts(filtered);
 };
```

UIDisplay:

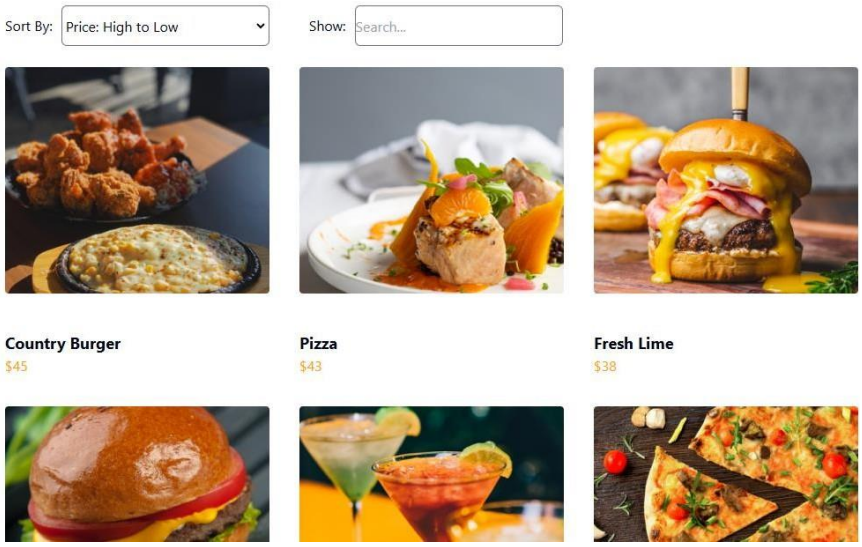


2. **PriceFiltering:**
   - Addoptionstosortproductsbypricein**ascending**or**descending**order.
   - Combinethepricefilterwiththesearchbarandcategoryfilterforseamless interaction

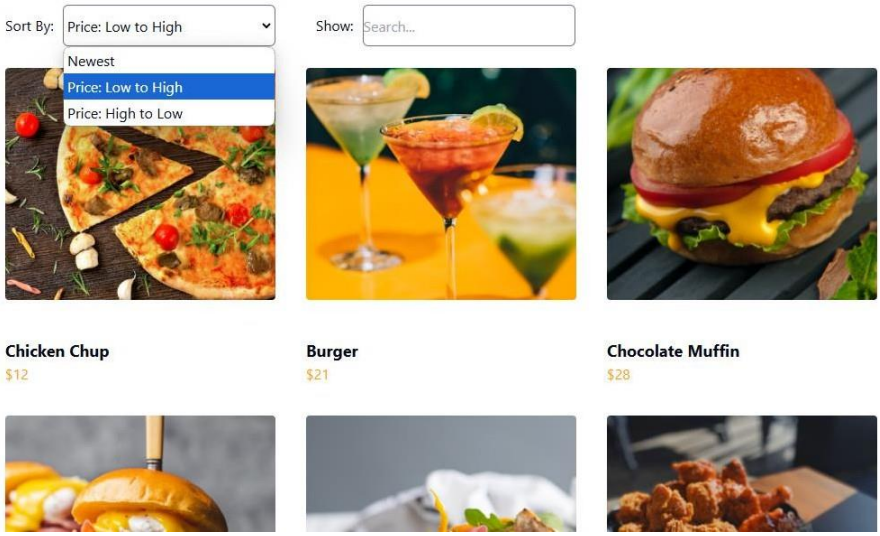
```
// Handle sorting
  const handleSort = (event: React.ChangeEvent<HTMLSelectElement>) => {
    const sortValue = event.target.value;
    setSortOrder(sortValue);

    let sortedProduct = [...products];
    if (sortValue === "lowToHigh") {
      sortedProduct.sort((a, b) => a.price - b.price);
    } else if (sortValue === "highToLow") {
      sortedProduct.sort((a, b) => b.price - a.price);
    }
    setFilteredProducts(sortedProduct);
  };
```

***UIDisplay:***

- ***HighToLow:***



- ***LowToHigh:***




## FeaturesImplemented:

1. **SearchBar:**
   - Filtersproductsbynameortagsinrealtime.
2. **PriceFilter:**
   - Allowssortingproductsbyprice(lowtohighorhightolow).

---

## Step4:Cart Component

### Objective:

Tocreatea**CartComponent**thatdisplaystheitemsaddedtothecart,theirquantity,andthe total price of the cart dynamically.

---

## ImplementationPlan:

1. **StateManagement:**
   - Use**Reactstate**orastatemanagementlibrarylikeReduxforstoringcart data.
2. **Cart Data:**
   - Includedetailsforeachproductinthecart:
     - ProductName
     - Price
     - Quantity
   - Calculateanddisplaythe**totalprice**dynamicallybasedontheitemsinthecart.

3. **Cart Interactions:**
   - Allowuserstoincreaseordecreasethequantityofitems.
   - Automaticallyupdatethetotalpricewhenthequantitychanges.

```
// Handle Increment
const handleIncrement = () => {
  const newQuantity = quantity + 1;
  setQuantity(newQuantity);
  setCartPrice(newQuantity * product.price); // Update price
};
 // Handle Decrement
 const handleDecrement = () => {
  if (quantity > 1) {
    const newQuantity = quantity - 1;
    setQuantity(newQuantity);
    setCartPrice(newQuantity * product.price);
  }
};

function handleAddToCart() {
  const cartItem = {
    slug: product.slugs,
    title: product.name,
    img: product.imageUrl,
    price: product.price,
    quantity: 1,
  };

  dispatch(addToCart(cartItem));
}
```

**FeaturesImplemented:**

1. **DynamicItemDisplay:**
   - Eachiteminthecartisdisplayedwithitsname,price,andquantity.
   - Subtotalforeachitemisdynamicallycalculated.
2. **Quantity Update:**
   - Buttonstoincrease(+)ordecrease(−)thequantityofanitem.
   - Quantitycannotgobelow1.
3. **TotalPrice Calculation:**
   - Thetotalpriceupdatesdynamicallyasitemsareaddedorquantitiesarechanged.
4. **Remove Item:**
   - Userscanremoveindividualitemsfromthecart.

## Step6:NotificationsComponent

### Objective:

Tocreatea**NotificationsComponent**thatdisplaysreal-timealertsforuseractions,suchas adding items to the cart, encountering errors, or completing a successful purchase.

---

## ImplementationPlan:

1. **Real-TimeAlerts:**
   - Use**toastnotifications**or**modalwindows**todisplayalerts.
   - Displaynotificationsforactions like:
     - Itemaddedtothecart
     - Errors(e.g.,"Outofstock")
     - Successfulactions(e.g.,"Purchasecomplete")
2. **Integration:**
   - Triggernotificationsatappropriatemomentsintheapp,suchasaddingtothecart or completing a transaction.
3. **Libraries:**
   - Useapopularnotificationlibrarylike**react-toastify**orbuildacustomnotification system.

```
const handleNotification = () => {toast.success('🦄 Item was add in cart sucessfully', {
  position: "top-center",
  autoClose: 2000,
  hideProgressBar: false,
  closeOnClick: false,
  pauseOnHover: true,
  draggable: true,
  progress: undefined,
  theme: "light",
  transition: Bounce,
  });
}
```

## Conclusion

On **Day 4** of building dynamic frontend components for a marketplace, the focus was on creatingmodular,reusable,andresponsivecomponents.Thefollowingkeycomponentswere successfully implemented:

1. **ProductListingComponent:**
   - Dynamicallydisplayedproductsinagridlayoutwithdetailssuchasproductname, price, image, and stock status.
2. **ProductDetailComponent:**
   - BuiltindividualproductpagesusingdynamicroutinginNext.js,includingfieldslike product description, price, and image.
3. **SearchBarandFilters:**
   - Implementedfunctionalitytofilterproductsbynameortagsandaddedprice filters (high to low and low to high).
4. **Cart Component:**
   - Displayeditemsaddedtothecart,quantitymanagement,andtotalprice calculation with dynamic update