Building Dynamic Frontend Components for Hunar Bazar

1.0 Objective:

This documentation aims to Detail the architecture, workflows, and interactions of all components in the Hunar Bazar e-commerce platform and, explain the use of Next.js, Zustand, Sanity CMS, and Clerk for a modular, scalable frontend.

2.0 Key Learning outcomes:

- Built 15+ dynamic components using Next.js and Sanity CMS integration.
- Implemented state management for cart, via Zustand.
- .Applied responsive styling with Tailwind CSS and ShadcnUI for a polished UI/UX.
- Integrated real-world workflows like dynamic routing, API error handling, and modular design.

3.0 Key Components Developed:

I Below are the components built for Hunar Bazar:

3.1 Components of Product Listing System & Filtering by Categories :

1. ### ProductList Component:

Purpose: Renders a grid of products fetched from Sanity CMS

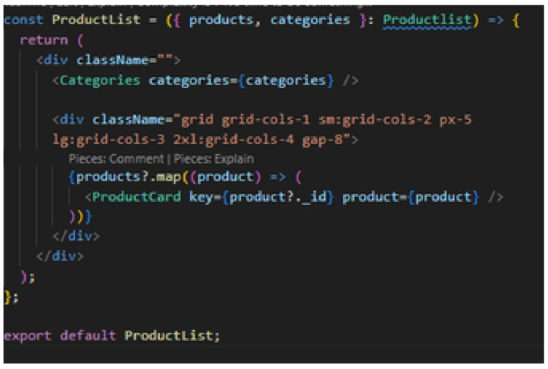
Key Features:

- Responsive grid layout (1 → 4 columns).
- Integrates category filtering.
- Maps product data to ProductCard components.

Workflow:

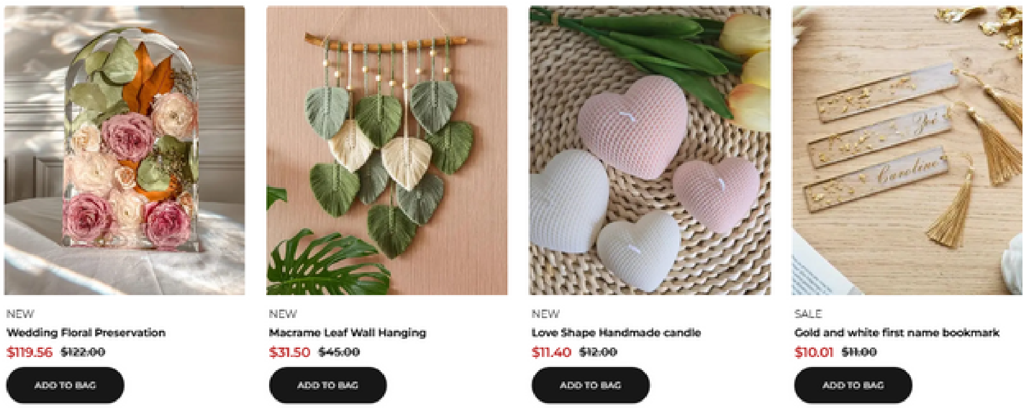
- Generates image URL using urlFor().
- Links to product detail page via slug
- Renders AddtoBag and PriceView components

Code Snippets:



```
const ProductList = ({ products, categories }: Productlist) => {
  return (
    <div className="">
      <Categories categories={categories} />

      <div className="grid grid-cols-1 sm:grid-cols-2 px-5
      lg:grid-cols-3 2xl:grid-cols-4 gap-8">
        Pieces: Comment | Pieces: Explain
        {products?.map((product) => (
          <ProductCard key={product?._id} product={product} />
        ))}
      </div>
    </div>
  );
};

export default ProductList;
```

Visual Representation:


NEW
Wedding Floral Preservation
$119.56 $122.00
ADD TO BAG
NEW
Macrame Leaf Wall Hanging
$31.50 $45.00
ADD TO BAG
NEW
Love Shape Handmade candle
$11.40 $12.00
ADD TO BAG
SALE
Gold and white first name bookmark
$10.01 $11.00
ADD TO BAG

2. ProductCard Component:

Purpose: Displays individual product details.

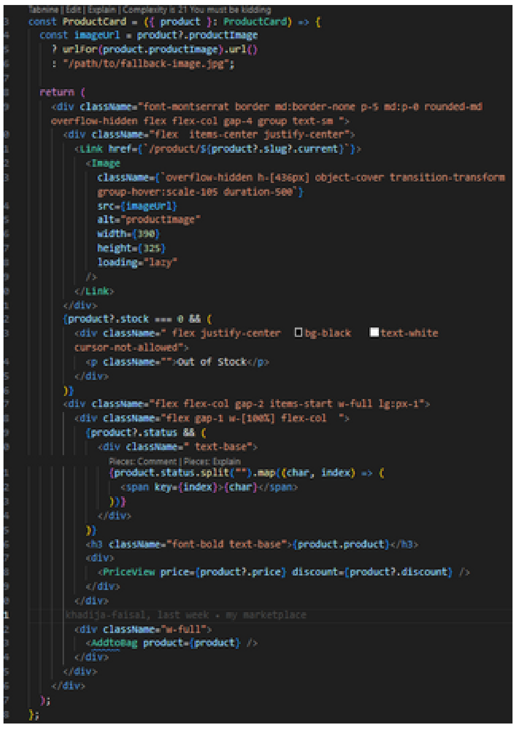
Key Features:

- Dynamic image loading from Sanity.
- Price display with discounts.
- Stock status indicator.

Workflow:

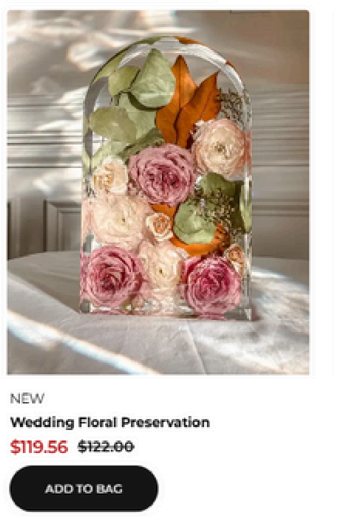
- Generates image URL using urlFor().
- Links to product detail page via slug
- Renders AddtoBag and PriceView components

Code Snippets:



```
Tabnine | Edit | Explain | Complexity is 21 You must be kidding
const ProductCard = ({ product }: ProductCard) => {
  const imageUrl = product?.productImage
    ? urlFor(product.productImage).url()
    : "/path/to/fallback-image.jpg";

  return (
    <div className="font-montserrat border md:border-none p-5 md:p-0 rounded-md
    overflow-hidden flex flex-col gap-4 group text-sm ">
      <div className="flex  items-center justify-center">
        <Link href={`/product/${product?.slug?.current}`}>
          <Image
            className={`overflow-hidden h-[436px] object-cover transition-transform
            group-hover:scale-105 duration-500`}
            src={imageUrl}
            alt="productImage"
            width={390}
            height={325}
            loading="lazy"
          />
        </Link>
      </div>
      {product?.stock === 0 && (
        <div className=" flex justify-center  ■bg-black  □text-white
        cursor-not-allowed">
          <p className="">Out of Stock</p>
        </div>
      )}
      <div className="flex flex-col gap-2 items-start w-full lg:px-1">
        <div className="flex gap-1 w-[100%] flex-col  ">
          {product?.status && (
            <div className=" text-base">
              Pieces: Comment | Pieces: Explain
              {product.status.split("").map((char, index) => (
                <span key={index}>{char}</span>
              ))}
            </div>
          )}
          <h3 className="font-bold text-base">{product.product}</h3>
          <div>
            <PriceView price={product?.price} discount={product?.discount} />
          </div>
        </div>
        khadija-faisal, last week • my marketplace
        <div className="w-full">
          <AddtoBag product={product} />
        </div>
      </div>
    </div>
  );
};
```

Visual Representation:



3. Categories Component:

Purpose: Wrapper for category filtering dropdown.

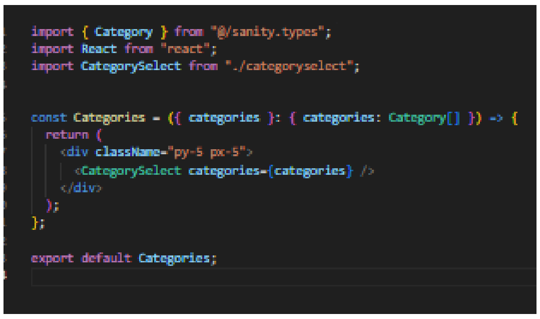
Key Features:

- Consistent padding/spacing.
- Passes Sanity CMS categories to CategorySelect.

Workflow:

- Receives categories prop from parent.
- Renders CategorySelect with categories.

Code Snippets:



```
import { Category } from "@/sanity.types";
import React from "react";
import CategorySelect from "./categoryselect";


const Categories = ({ categories }: { categories: Category[] }) => {
  return (
    <div className="py-5 px-5">
      <CategorySelect categories={categories} />
    </div>
  );
};

export default Categories;
```

4. CategorySelect Component:

Purpose: Wrapper for category filtering dropdown.

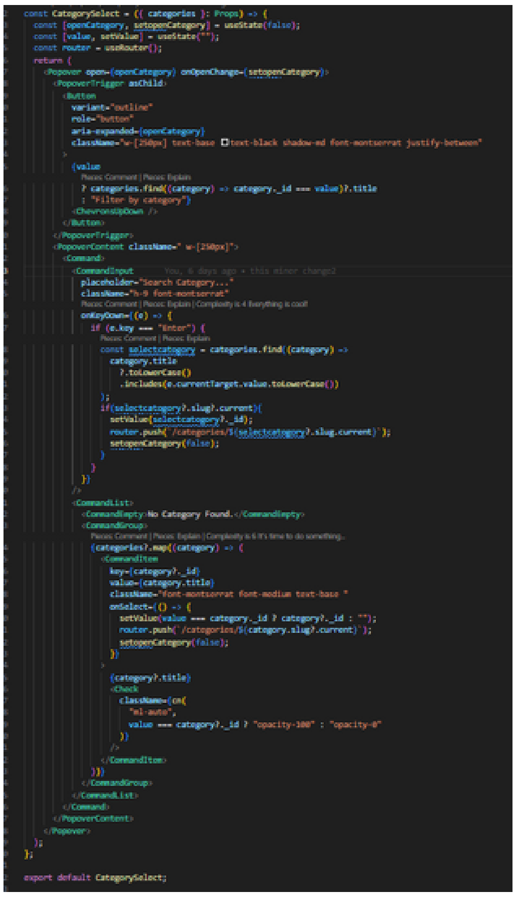
Key Features:

- Searchable list of categories.
- Dynamic routing on selection.

Workflow:

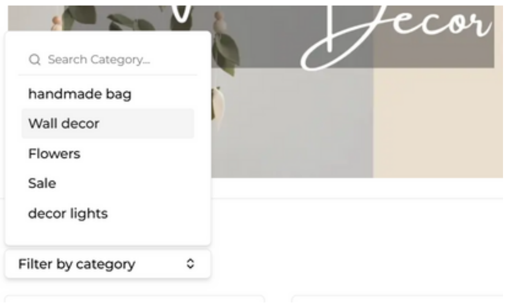
- Opens a ShadcnUI Popover on click.
- Filters categories via search input.
- Updates URL using Next.js router.push().

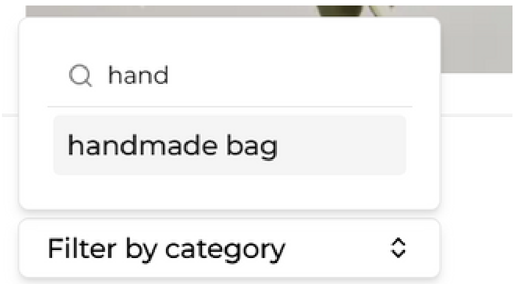
Code Snippets:



```
const CategorySelect = ({ categories }: Props) => {
  const [openCategory, setopenCategory] = useState(false);
  const [value, setValue] = useState("");
  const router = useRouter();
  return (
    <Popover open={openCategory} onOpenChange={setopenCategory}>
      <PopoverTrigger asChild>
        <Button
          variant="outline"
          role="button"
          aria-expanded={openCategory}
          className="w-[250px] text-base text-black shadow-md font-montserrat justify-between"
        >
          {value
            Pieces: Comment | Pieces: Explain
            ? categories.find((category) => category._id === value)?.title
            : "Filter by category"}
          <ChevronsUpDown />
        </Button>
      </PopoverTrigger>
      <PopoverContent className=" w-[250px]">
        <Command>
          <CommandInput        You, 6 days ago • this minor change2
            placeholder="Search Category..."
            className="h-9 font-montserrat"
            Pieces: Comment | Pieces: Explain | Complexity is 4 Everything is cool!
            onKeyDown={(e) => {
              if (e.key === "Enter") {
                Pieces: Comment | Pieces: Explain
                const selectcategory = categories.find((category) =>
                  category.title
                    ?.toLowerCase()
                    .includes(e.currentTarget.value.toLowerCase())
                );
                if(selectcategory?.slug?.current){
                  setValue(selectcategory?._id);
                  router.push(`/categories/${selectcategory?.slug.current}`);
                  setopenCategory(false);
                }
              }
            }}
          />
          <CommandList>
            <CommandEmpty>No Category Found.</CommandEmpty>
            <CommandGroup>
              Pieces: Comment | Pieces: Explain | Complexity is 6 It's time to do something...
              {categories?.map((category) => (
                <CommandItem
                  key={category?._id}
                  value={category.title}
                  className="font-montserrat font-medium text-base "
                  onSelect={() => {
                    setValue(value === category._id ? category?._id : "");
                    router.push(`/categories/${category.slug?.current}`);
                    setopenCategory(false);
                  }}
                >
                  {category?.title}
                  <Check
                    className={cn(
                      "ml-auto",
                      value === category?._id ? "opacity-100" : "opacity-0"
                    )}
                  />
                </CommandItem>
              ))}
            </CommandGroup>
          </CommandList>
        </Command>
      </PopoverContent>
    </Popover>
  );
};

export default CategorySelect;
```

Visual Representation:





3.2 Product Quantity with price adjustment & Cart Interaction for Products:

1. AddtoBag Button Component:

Purpose: Handles cart interactions for a product.

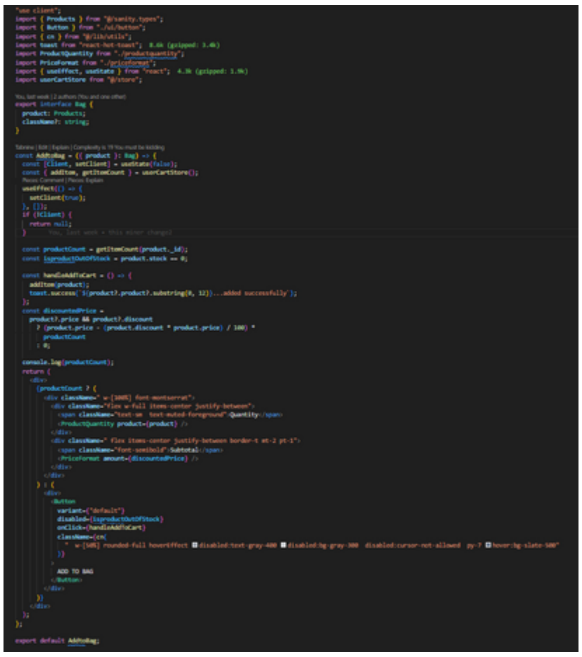
Key Features:

- Zustand state management.
- Toast notifications.
- Quantity adjustment controls.

Workflow:

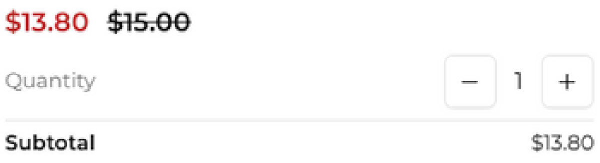
- Check product stock status.
- Updates cart state via addItem().
- Switches to ProductQuantity after adding to the cart.

Code Snippets:



```
"use client";
import { Products } from "@/sanity.types";
import { Button } from "./ui/button";
import { cn } from "@/lib/utils";
import toast from "react-hot-toast";  8.6k (gzipped: 3.4k)
import ProductQuantity from "./productquantity";
import PriceFormat from "./priceformat";
import { useEffect, useState } from "react";  4.3k (gzipped: 1.9k)
import userCartStore from "@/store";

You, last week | 2 authors (You and one other)
export interface Bag {
  product: Products;
  className?: string;
}

Tabnine | Edit | Explain | Complexity is 11 You must be kidding
const AddtoBag = ({ product }: Bag) => {
  const [client, setClient] = useState(false);
  const { addItem, getItemCount } = userCartStore();
  Pieces: Comment | Pieces: Explain
  useEffect(() => {
    setClient(true);
  }, []);
  if (!client) {
    return null;
  }      You, last week • this minor change2

  const productCount = getItemCount(product._id);
  const isproductOutOfStock = product.stock == 0;

  const handleAddToCart = () => {
    addItem(product);
    toast.success(`${product?.product?.substring(0, 12)}...added successfully`);
  };
  const discountedPrice =
    product?.price && product?.discount
      ? (product.price - (product.discount * product.price) / 100) *
        productCount
      : 0;

  console.log(productCount);
  return (
    <div>
      {productCount ? (
        <div className=" w-[100%] font-montserrat">
          <div className="flex w-full items-center justify-between">
            <span className="text-sm  text-muted-foreground">Quantity</span>
            <ProductQuantity product={product} />
          </div>
          <div className=" flex items-center justify-between border-t mt-2 pt-1">
            <span className="font-semibold">Subtotal</span>
            <PriceFormat amount={discountedPrice} />
          </div>
        </div>
      ) : (
        <div>
          <Button
            variant={"default"}
            disabled={isproductOutOfStock}
            onClick={handleAddToCart}
            className={cn(
              "  w-[50%] rounded-full hoverEffect ■disabled:text-gray-400 ■disabled:bg-gray-300  disabled:cursor-not-allowed  py-7 ■hover:bg-slate-500"
            )}
          >
            ADD TO BAG
          </Button>
        </div>
      )}
    </div>
  );
};

export default AddtoBag;
```

Visual Representation:



2. PriceView and PriceFormat Components:

Purposes: Displays discounted and original prices, formats prices into USD currency.

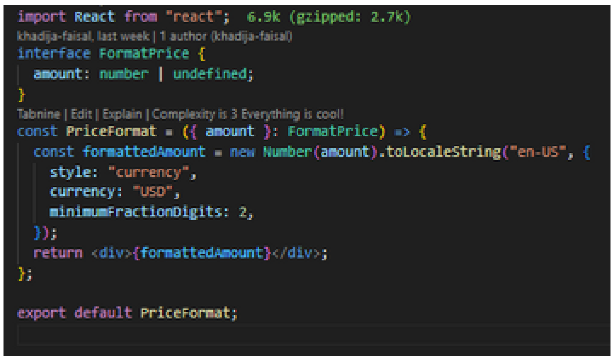
Key Features:

- Strikethrough for original price.
- Red text for a discounted price.
- Reusable across components.
- Locale-aware formatting.

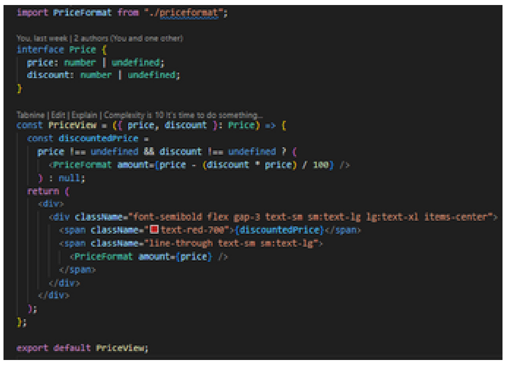
Workflow:

- Calculates discounted price.
- Renders prices using PriceFormat.
- Accepts amount prop.
- Converts number to currency string (e.g., $49.99).

Code Snippets:



```
import React from "react";  6.9k (gzipped: 2.7k)
khadija-faisal, last week | 1 author (khadija-faisal)
interface FormatPrice {
  amount: number | undefined;
}
Tabnine | Edit | Explain | Complexity is 3 Everything is cool!
const PriceFormat = ({ amount }: FormatPrice) => {
  const formattedAmount = new Number(amount).toLocaleString("en-US", {
    style: "currency",
    currency: "USD",
    minimumFractionDigits: 2,
  });
  return <div>{formattedAmount}</div>;
};

export default PriceFormat;
```



```
import PriceFormat from "./priceformat";

You, last week | 2 authors (You and one other)
interface Price {
  price: number | undefined;
  discount: number | undefined;
}

Tabnine | Edit | Explain | Complexity is 10 It's time to do something...
const PriceView = ({ price, discount }: Price) => {
  const discountedPrice =
    price !== undefined && discount !== undefined ? (
      <PriceFormat amount={price - (discount * price) / 100} />
    ) : null;
  return (
    <div>
      <div className="font-semibold flex gap-3 text-sm sm:text-lg lg:text-xl items-center">
        <span className="🟥 text-red-700">{discountedPrice}</span>
        <span className="line-through text-sm sm:text-lg">
          <PriceFormat amount={price} />
        </span>
      </div>
    </div>
  );
};

export default PriceView;
```

3. ProductQuantity Component:

Purposes: Adjusts item quantity in the cart

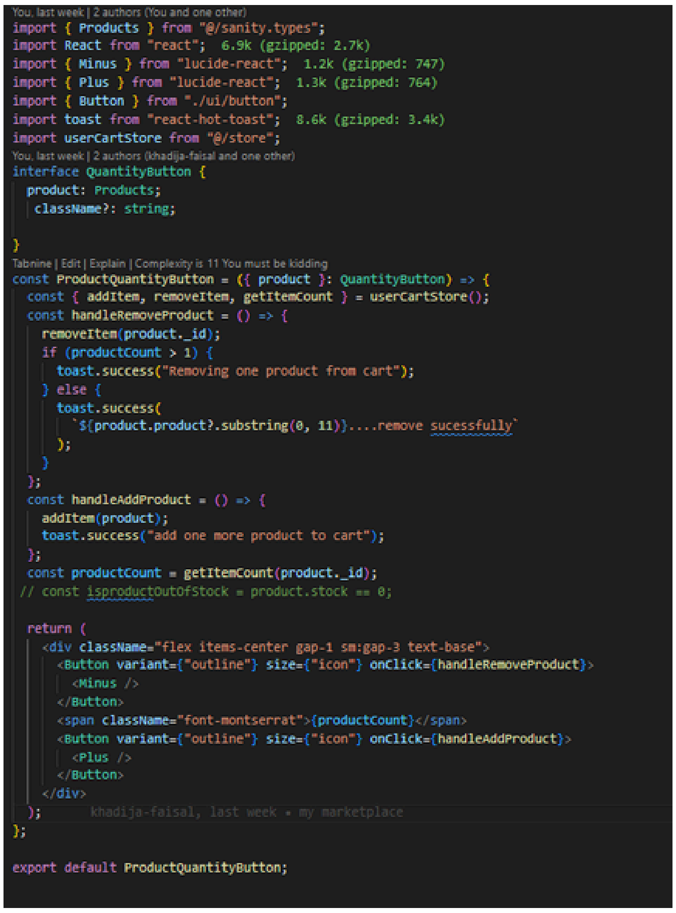
Key Features:

- "+" and "-" buttons.
- Zustand state updates.

Workflow:

- Calls addItem()/removeItem() on click.
- Shows live count with toast feedback.

Code Snippets:



```
You, last week | 2 authors (You and one other)
import { Products } from "@/sanity.types";
import React from "react";  6.9k (gzipped: 2.7k)
import { Minus } from "lucide-react";  1.2k (gzipped: 747)
import { Plus } from "lucide-react";  1.3k (gzipped: 764)
import { Button } from "./ui/button";
import toast from "react-hot-toast";  8.6k (gzipped: 3.4k)
import userCartStore from "@/store";
You, last week | 2 authors (khadija-faisal and one other)
interface QuantityButton {
  product: Products;
   className?: string;

}
Tabnine | Edit | Explain | Complexity is 11 You must be kidding
const ProductQuantityButton = ({ product }: QuantityButton) => {
  const { addItem, removeItem, getItemCount } = userCartStore();
  const handleRemoveProduct = () => {
    removeItem(product._id);
    if (productCount > 1) {
      toast.success("Removing one product from cart");
    } else {
      toast.success(
        `${product.product?.substring(0, 11)}....remove sucessfully`
      );
    }
  };
  const handleAddProduct = () => {
    addItem(product);
    toast.success("add one more product to cart");
  };
  const productCount = getItemCount(product._id);
 // const isproductOutOfStock = product.stock == 0;

  return (
    <div className="flex items-center gap-1 sm:gap-3 text-base">
      <Button variant={"outline"} size={"icon"} onClick={handleRemoveProduct}>
        <Minus />
      </Button>
      <span className="font-montserrat">{productCount}</span>
      <Button variant={"outline"} size={"icon"} onClick={handleAddProduct}>
        <Plus />
      </Button>
    </div>
  );     khadija-faisal, last week • my marketplace
};

export default ProductQuantityButton;
```

3.3 Header, footer, CartIcon Components:

1. Header & CartIcon Component:

Purposes: Main navigation bar, Shows cart item count in the header.

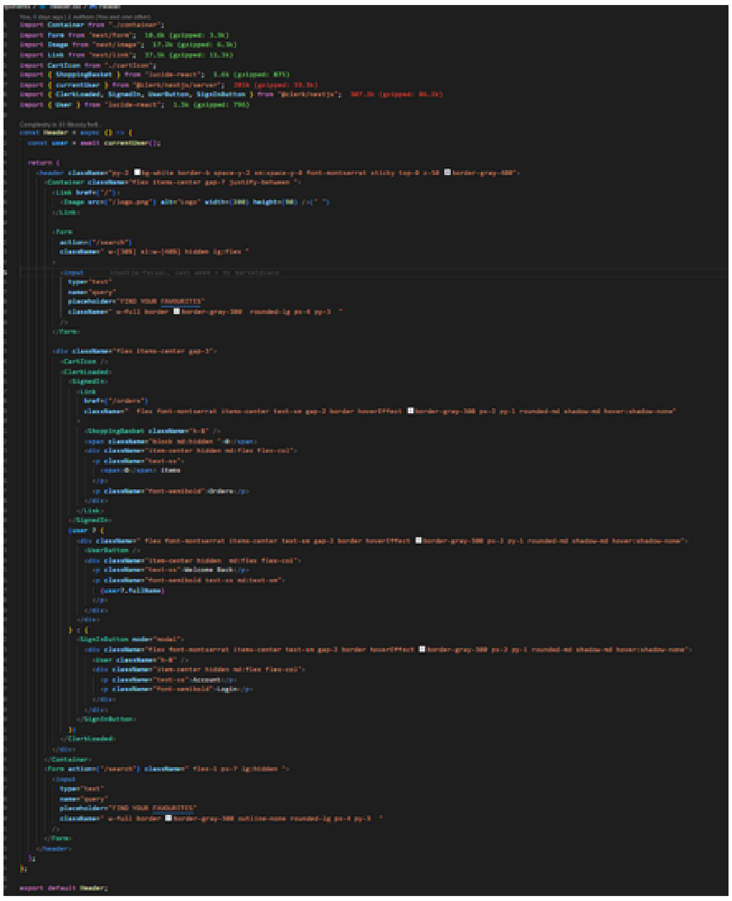
Key Features:

- Clerk authentication.
- Responsive search bar.
- Responsive design (icon + text).
- Zustand state synchronization.

Workflow:

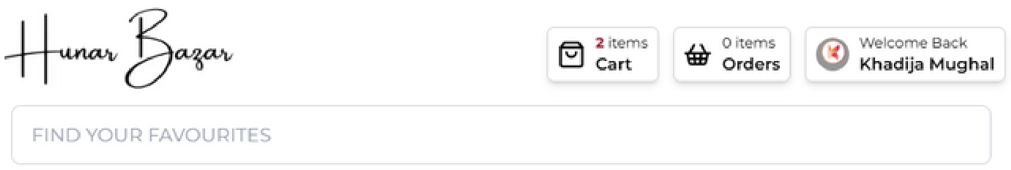
- Displays logo, search bar, cart icon, and auth buttons.
- Uses ClerkLoaded to conditionally render auth state.
- Fetches cart items via getGroupedItems().
- Updates UI on cart state changes.

Code Snippets:



```
You, 4 days ago | 2 authors (You and one other)
import Container from "./container";
import Form from "next/form";  10.6k (gzipped: 3.3k)
import Image from "next/image";  17.2k (gzipped: 6.3k)
import Link from "next/link";  37.5k (gzipped: 13.3k)
import CartIcon from "./cartIcon";
import { ShoppingBasket } from "lucide-react";  1.4k (gzipped: 875)
import { currentUser } from "@clerk/nextjs/server";  381k (gzipped: 55.5k)
import { ClerkLoaded, SignedIn, UserButton, SignInButton } from "@clerk/nextjs";  307.2k (gzipped: 86.2k)
import { User } from "lucide-react";  1.3k (gzipped: 790)

Complexity is 11 Bloody hell...
const Header = async () => {
  const user = await currentUser();

  return (
    <header className="py-2 bg-white border-b space-y-2 md:space-y-0 font-montserrat sticky top-0 z-50 border-gray-400">
      <Container className="flex items-center gap-7 justify-between ">
        <Link href={"/"}>
          <Image src={"/logo.png"} alt="logo" width={300} height={90} /*{" "}
        </Link>

        <Form
          action={"/search"}
          className=" w-[305] xl:w-[405] hidden lg:flex "
        >
          <input        khadija-faisal, last week • my marketplace
            type="text"
            name="query"
            placeholder="FIND YOUR FAVOURITES"
            className=" w-full border border-gray-300  rounded-lg px-4 py-3  "
          />
        </Form>

        <div className="flex items-center gap-3">
          <CartIcon />
          <ClerkLoaded>
            <SignedIn>
              <Link
                href={"/orders"}
                className="  flex font-montserrat items-center text-sm gap-2 border hoverEffect border-gray-300 px-2 py-1 rounded-md shadow-md hover:shadow-none"
              >
                <ShoppingBasket className="h-8" />
                <span className="block md:hidden ">0</span>
                <div className="item-center hidden md:flex flex-col">
                  <p className="text-xs">
                    <span>0</span> items
                  </p>
                  <p className="font-semibold">Orders</p>
                </div>
              </Link>
            </SignedIn>
            {user ? (
              <div className=" flex font-montserrat items-center text-sm gap-2 border hoverEffect border-gray-300 px-2 py-1 rounded-md shadow-md hover:shadow-none">
                <UserButton />
                <div className="item-center hidden  md:flex flex-col">
                  <p className="text-xs">Welcome Back</p>
                  <p className="font-semibold text-xs md:text-sm">
                    {user?.fullName}
                  </p>
                </div>
              </div>
            ) : (
              <SignInButton mode="modal">
                <div className="flex font-montserrat items-center text-sm gap-2 border hoverEffect border-gray-300 px-2 py-1 rounded-md shadow-md hover:shadow-none">
                  <User className="h-8" />
                  <div className="item-center hidden md:flex flex-col">
                    <p className="text-xs">Account</p>
                    <p className="font-semibold">Login</p>
                  </div>
                </div>
              </SignInButton>
            )}
          </ClerkLoaded>
        </div>
      </Container>
      <Form action={"/search"} className=" flex-1 px-7 lg:hidden ">
        <input
          type="text"
          name="query"
          placeholder="FIND YOUR FAVOURITES"
          className=" w-full border border-gray-300 outline-none rounded-lg px-4 py-3  "
        />
      </Form>
    </header>
  );
};

export default Header;
```

Visual Representation:



2. Footer Component:

Purposes: Site-wide footer with contact/social links.

Key Features:

- 4-column responsive layout.
- Newsletter signup form.

Workflow:

- Static content with Tailwind styling.
- Social media icons with SVG

3.4 Fallback Components:

EmptyCart, CartAccessDenied & Loader Component:

Purposes: Empty cart state UI, Blocks unauthorized cart access, Full-screen loading spinner.

Key Features:

- Animated shopping bag icon.
- "Start Shopping" CTA button.
- Clerk authentication modal.
- Branded card design.
- Puff animation from react-loader-spinner.
- Centered layout.

Workflow:

- Triggers when cart is empty.
- Uses Framer Motion for hover effects.
- Renders when unauthenticated users visit /cart.
- Prompts login via SignInButton.
- Displays during data fetching/API calls.
- Uses aria-label for accessibility.

3.5 Banner Component:

Purposes: Displays promotional carousel.

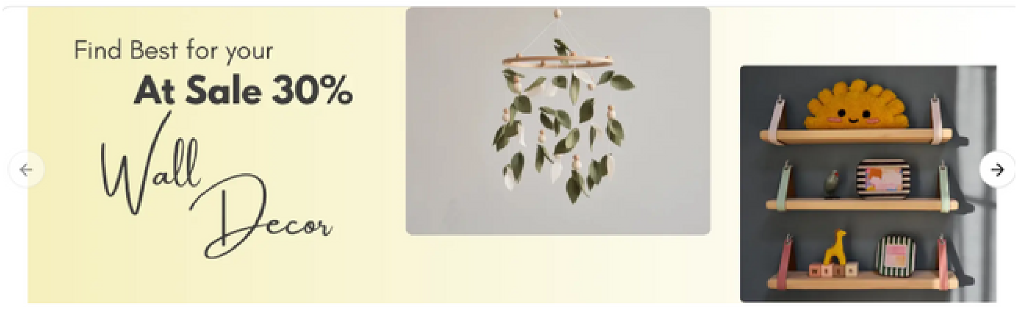
Key Features:

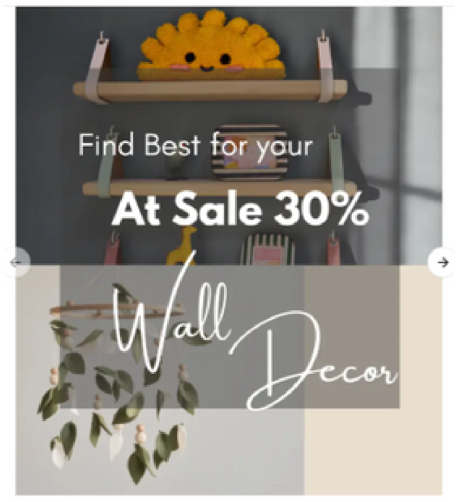
- ShadcnUI Carousel component.
- Responsive images (mobile/desktop).

Workflow:

- Fetches banners from Sanity CMS.
- Switches images based on screen size.

Visual Representation:





4.0 Best Practice Implemented on Components:

Components:	**Key Practice:**	**Impact:**
header.tsx	Clerk auth	Secure user sessions
AddtoBag.tsx	Cart persistence + toasts	Retains cart data + user feedback
CategorySelect.tsx	Keyboard navigation	Inclusive UX for all users
PriceFormat.tsx	Reusable utility	Consistent pricing across app
ProductCard.tsx	Lazy-loading + SEO	Faster loads + better SEO ranking

5.0 Conclusion:

Hunar Bazar showcases a modern e-commerce frontend built with Next.js, Zustand, and Sanity CMS, emphasizing modularity, performance, and user-centric design.

5.1 Key Wins:

- **State Management**: Cart persistence via Zustand + localStorage (AddtoBag.tsx).
- **Performance**: Lazy-loaded images (ProductCard.tsx), reusable utilities (PriceFormat.tsx).
- **Accessibility**: Keyboard navigation (CategorySelect.tsx), semantic HTML (Footer.tsx).
- **Security**: Clerk authentication for protected routes (CartAccessDenied.tsx).

5.2 Outcomes:

- **User Experience**: Toast notifications, responsive grids, and guided empty states.
- **Maintainability**: Decoupled components (e.g., Categories.tsx, ProductList.tsx).

5.3 Next Steps:

- Integrate Stripecheckout.
- Add debounced search to CategorySelect.tsx.
- Build an admin dashboard for Sanity CMS.

Hunar Bazar is a scalable, production-ready foundation for future expansion, balancing technical rigor with intuitive design.