"Day 4 - Dynamic Frontend Components – Food Tuck"

Code Snippets of [id]/page.tsx

What I create here:

- Dynamic routing
- Dummy effect of notification when click on wish list and add to cart
- Related food items effect by category
- Product detail

```
"use client";
import React from "react";
import { client } from "@/sanity/lib/client";
import { ShopItems } from "@/app/component/RouteHead/RouteHead";
import Header1 from "@/app/component/Header/Header1";
import { Inter } from "next/font/google";
import StarRating from "@/app/component/star/star";
import { AmountSetterShop } from "@/app/component/counter/counter";
import { Button } from "@/components/ui/button";
import { PiBagLight } from "react-icons/pi";
import { FaHeart } from "react-icons/fa";
import { Foods15 } from "@/sanity/lib/queries";
import Link from "next/link";


const inter = Inter({ weight: ["400", "700"], subsets: ["latin"] });

type ItemParams = {
  params: Promise<{ _id: string }>;
};

type Food = {
  _id: string;
  name: string;
  category: string;
  price: number;
  originalPrice: number;
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
  tags: string[];
  imageUrl: string;
  description: string;
  available: true;
};

async function getItems(_id: string) {
  const Fetch = await client.fetch(
    `*[_type == "food" && _id == $id]{
    "_id": _id,
    name,
    description,
    price,
    category,
    originalPrice,
    tags,
    "imageUrl":image.asset -> url,
    available
}`,
    { id: _id }
  );



  if (!Fetch.length) {
    throw new Error(`No item found with the id "${_id}"`);
  }

  return Fetch[0]; // Return the single object
}

async function page({ params }: ItemParams) {
  const products: Food[] = await Foods15(0, 14);//get all products
  const itemsInfo:Food = await getItems((await params)._id); // Fetch  data using
the `id` from URL



  const showAlert = () => {
    alert("Item added to cart"); //just for alert effect
  };

  const addTOWishList = ()=>{

    alert("Item added to Wish list");//just for alert effect
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
  }


 //get similar items
  const relatedItems = products.filter((product) => product.category ===
itemsInfo.category);




  return (
    <div>
      <Header1 />
      <ShopItems />
      <div className="flex flex-col  justify-center items-center py-[150px] gap-
10">
        <div className="flex flex-row  justify-center items-center  gap-10">
        <div >
          <img
            src={itemsInfo.imageUrl}
            width={449}
            height={569}
            alt={itemsInfo.name}
            className="rounded-[6px]"
          />
        </div>
        <div className="w-[608px] flex flex-col gap-[16px]">
          <div className=" flex flex-col ">
            <div className=" py-2 px-1 rounded-[6px] text-white w-auto h-auto
flex">
              {itemsInfo.available ? (
                <div className="bg-[#FF9F0D] px-[6px] py-[7px] rounded-[6px]">
                  In stock
                </div>
              ) : (
                <div className="bg-[#595858] px-[6px] py-[7px] rounded-[6px]">
                  Stock Out
                </div>
              )}
            </div>
            <h1 className="font-helvetica font-bold text-[48px]">
              {itemsInfo.name}
            </h1>
            <p className="text-[#4F4F4F] text-[18px]" style={inter.style}>
              {itemsInfo.description}
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
          </p>
        </div>
        <hr />
        <div className=" flex flex-col gap-[8px] ">
          <p className="font-bold text-[32px] font-helvetica">
            {itemsInfo.price.toFixed(2)}$
          </p>{" "}
          {/*To make number like 50.00$ */}
          <div>
            <StarRating />
          </div>
          <div>
            {itemsInfo.tags.join(" / ")}
            {/* to make tags like "food / cheezy / spicy" */}
          </div>
          <div className="flex flex-row gap-[16px]">
            <div>
              <AmountSetterShop />
            </div>
            <div>
              <Button
                variant="primary"
                size="none"
                className="text-white w-auto h-auto py-2 px-1"
                onClick={showAlert}
              >
                <PiBagLight /> Add to cart
              </Button>
            </div>
          </div>
          <div className="text-[18px] text-[#4F4F4F] flex flex-row gap-[8px]">
            <p>Add to Wishlist</p>
            <Button
              variant="none"
              size="none"
              className="focus:text-red-700 text-gray-400"
              onClick={addTOWishList}
            >
              <FaHeart />
            </Button>
          </div>
        </div>
      </div>
      </div>
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
        <div className="flex flex-col  justify-center items-start px-[50px] gap-
[10px]">
          <h4 className="text-black font-bold font-helvetica text-[32px]">Similar
Products</h4>
          <div className="flex flex-row  justify-center items-center  gap-5">
            {relatedItems.map((food)=>(
              <div
              key={food._id}
              className="w-auto flex flex-col items-start justify-between gap-
[8px]"
            >
              <div>
                <img
                  src={food.imageUrl}
                  width={212}
                  height={267}
                  alt={food.name}
                />
              </div>
              <div className="w-full flex flex-col items-start justify-between
gap-[8px]">
                <h3 className="font-bold text-[18px]">{food.name}</h3>
                <div className="w-auto flex flex-row items-start justify-between
gap-[16px]">
                  <div className="text-[#FF9F0D]">
                    ${food.originalPrice.toFixed(2)}
                  </div>
                  <div>
                    {food.price ? (
                      <div className="text-[#828282] line-through">
                        ${food.price.toFixed(2)}
                      </div>
                    ) : (
                      <div className="hidden"></div>
                    )}
                  </div>
                </div>
              </div>
              <div className="flex flex-row w-full justify-between items-center">
                <Button
                  variant="primary"
                  size="none"
                  className="py-2 px-2 hover:bg-[#ffaf37] drop-shadow-
[3px_3px_1px_rgba(189,188,187,0.5)] active:drop-shadow-none"
                >
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
                  <Link
                    href={`/Shop/${food._id}`}
                    className="text-[14px]"
                  >
                    Order Now!
                  </Link>
                </Button>
                <div className=" py-2 px-1 rounded-[6px] text-white w-auto h-auto
flex">
                  {food.available ? (
                    <div className="bg-[#FF9F0D] px-[6px] py-[7px] rounded-
[6px]">
                      In stock
                    </div>
                  ) : (
                    <div className="bg-[#595858] px-[6px] py-[7px] rounded-
[6px]">
                      Stock Out
                    </div>
                  )}
                </div>
              </div>
            </div>
            ))}
          </div>
        </div>

      </div>

    </div>
  );
}

export default page;
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

Code Snippets of Food/page.tsx

What I create here:

- Product listing
- Category filter
- Loading effect on product list

```
"use client";
import { useState, useEffect } from "react";
import Header1 from "../component/Header/Header1";
import { Shop } from "../component/RouteHead/RouteHead";
import { Inter } from "next/font/google";
import { Checkbox } from "@/components/ui/checkbox";
import { Button } from "@/components/ui/button";
import { IoArrowForwardCircleOutline } from "react-icons/io5";
import { Slider } from "@/components/ui/slider";
import { Foods15 } from "@/sanity/lib/queries";
import Link from "next/link";
import Searchbar from "../component/Searchbar/searchbar";


const inter = Inter({ weight: ["400", "700"], subsets: ["latin"] });

type Food = {
  _id: string;
  name: string;
  category: string;
  price: number;
  originalPrice: number;
  tags: string[];
  imageUrl: string;
  description: string;
  available: true;
};


const filters = [
  "Drink",
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
  "Dessert",
  "Sandwich",
  "Main Course",
  "Appetizer",
  "Burger",
  "Pizza",
  "Non Veg",
];


function ShopItems() {
  const [selectedFilter, setSelectedFilter] = useState<string[]>([]); //check box
  const [filteredItems, setFilteredItems] = useState<Food[]>([]); //check box
  const [loading, setLoading] = useState(false); //loader


  //for check box
  const HandleFilterButtonWhenClick = (selectCatagory: string) => {
    if (selectedFilter.includes(selectCatagory)) {
      const filter = selectedFilter.filter((e) => e !== selectCatagory); //remove
items that is not equal to selected check box
      setSelectedFilter(filter);
    } else {
      setSelectedFilter([...selectedFilter, selectCatagory]);
    }
  };

  useEffect(() => {
    //for check box
    const fetchData = async () => {
      const products:Food[] = await Foods15(0,14);
      console.log(products)
      filterItems(products);
    };

    fetchData();

    //create loader
    setLoading(true);
    setTimeout(() => {
      console.log("this is running before component load");
      setLoading(false);
    }, 5000);
  }, [selectedFilter]);
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
  const filterItems = (products: Food[]) => {
    if (selectedFilter.length > 0) {
      const items = selectedFilter.map((selectedCatagory) => {
        const template = products.filter(
          (item) => item.category === selectedCatagory //separate and find
catagory
        );
        return template;
      });
      setFilteredItems(items.flat()); // set all catagory in one array
    } else {
      setFilteredItems([...products]);
    }
  };
  const load = (
    <div className="flex justify-center items-center h-screen lg:w-[984px] w-
auto">
      <div className="border-solid border-[#FF9F0D] border-b-[5px] border-t-[3px]
w-auto h-auto animate-spin rounded-full duration-500 ">
        {" "}
        <div className=" m-3 flex justify-center items-center border-solid
border-black border-[5px] w-[50px] h-[50px] animate-spin rounded-full duration-
500  "></div>
      </div>
    </div>
  );

  return (
    <div>
      <Header1 />
      <Shop />

      <div className="flex md:flex-row flex-col-reverse gap-[16px] justify-center
py-[150px]">
        {loading ? (
          load
        ) : (
          <div
            className="flex flex-col justify-center items-center gap-[20px]   "
            style={inter.style}
          >
            <div className="flex flex-col items-start gap-[32px]">
              <div className="flex lg:flex-row flex-col justify-center gap-
[32px]">
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
                <form className="flex flex-row gap-[16px] justify-center items-
center">
                  <label>Sort By :</label>
                  <select className="border-[#E0E0E0] border-[1px] border-solid
rounded-[6px] w-[236px] h-[46px] text-[#E0E0E0] focus:text-black">
                    <option defaultValue="Newest">Newest </option>
                    <option>Oldest</option>
                    <option>Popular</option>
                    <option>Latest</option>
                  </select>
                </form>
                <form className="flex flex-row gap-[16px] justify-center items-
center">
                  <label>Show :</label>
                  <select className="border-[#E0E0E0] border-[1px] border-solid
rounded-[6px] w-[236px] h-[46px] text-[#E0E0E0] focus:text-black">
                    <option defaultValue="Default">Default </option>
                    <option>Food Tuck Offical</option>
                  </select>
                </form>
              </div>
              <div className="grid lg:grid-cols-3  lg:gap-[32px] grid-col-1 gap-
[16px]">
                {filteredItems.map((food) => (
                  <div
                    key={food._id}
                    className="w-auto flex flex-col items-start justify-between
gap-[8px]"
                  >
                    <div>
                      <img
                        src={food.imageUrl}
                        width={312}
                        height={267}
                        alt={food.name}
                      />
                    </div>
                    <div className="w-full flex flex-col items-start justify-
between gap-[8px]">
                      <h3 className="font-bold text-[18px]">{food.name}</h3>
                      <div className="w-auto flex flex-row items-start justify-
between gap-[16px]">
                        <div className="text-[#FF9F0D]">
                          ${food.originalPrice.toFixed(2)}
                        </div>
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
                        <div>
                          {food.price ? (
                            <div className="text-[#828282] line-through">
                              ${food.price.toFixed(2)}
                            </div>
                          ) : (
                            <div className="hidden"></div>
                          )}
                        </div>
                      </div>
                    </div>
                    <div className="flex flex-row w-full justify-between items-
center">
                      <Button
                        variant="primary"
                        size="none"
                        className="py-2 px-2 hover:bg-[#ffaf37] drop-shadow-
[3px_3px_1px_rgba(189,188,187,0.5)] active:drop-shadow-none"
                      >
                        <Link
                          href={`/Shop/${food._id}`}
                          className="text-[14px]"
                        >
                          Order Now!
                        </Link>
                      </Button>
                      <div className=" py-2 px-1 rounded-[6px] text-white w-auto
h-auto flex">
                        {food.available ? (
                          <div className="bg-[#FF9F0D] px-[6px] py-[7px] rounded-
[6px]">
                            In stock
                          </div>
                        ) : (
                          <div className="bg-[#595858] px-[6px] py-[7px] rounded-
[6px]">
                            Stock Out
                          </div>
                        )}
                      </div>
                    </div>
                  </div>
                ))}
                {/* <Cards/>  */}
              </div>
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
            </div>
          </div>
        )}
        <div className="border-solid border-[#F2F2F2] border-[1px] rounded-[6px]
px-[16px] py-[32px] h-auto flex flex-col gap-[32px]">
          <div>
            <Searchbar />
          </div>
          <div className="flex flex-col  md:gap-[32px] gap-[16px]">
            <div>
              <h6 className="font-bold font-helvetica text-[20px] ">
                Category
              </h6>
            </div>
            <div className="md:flex flex-col grid sm:grid-cols-4 grid-cols-2 gap-
[16px]">
              {filters.map((catagory,i) => (
                <div key={i} className="flex flex-row gap-[8px]">
                  {" "}
                  <Checkbox
                    id="terms"
                    className=""
                    onClick={() => HandleFilterButtonWhenClick(catagory)}
                  />
                  <span className="text-sm text-[#333333]">{catagory}</span>
                </div>
              ))}
            </div>
          </div>
          <div className="bg-[url('/adofshop.svg')] py-[32px] px-[16px] flex
flex-col justify-between text-white h-[286px]">
            <div className="flex flex-col gap-[16px]">
              <div className="flex flex-col gap-[8px]">
                <p className="font-bold" style={inter.style}>
                  Perfect Taste
                </p>
                <h6 className="font-bold font-helvetice text-[20px]">
                  Classic Restaurant
                </h6>
              </div>

              <p className="font-bold text-[#FF9F0D]" style={inter.style}>
                45.00$
              </p>
            </div>
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
            <button
              className="flex flex-row gap-[8px] items-center"
              style={inter.style}
            >
              <p>Shop Now</p>{" "}
              <IoArrowForwardCircleOutline className="w-[24px] h-[24px] " />
            </button>
          </div>
          <div>
            <Slider defaultValue={[20]} max={100} step={1} />
          </div>
          <div></div>
          <div></div>
        </div>
      </div>
    </div>
  );
}

export default ShopItems;
```

Challenges faced and solutions implemented.

- Search bar and Pagination: get stuck at some point
- Wish list getting not array

Steps taken to build and integrate components

- ***Category checkbox***

1. First I create a Array for categories

```
const filters = [
  "Drink",
  "Dessert",
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
  "Sandwich",
  "Main Course",
  "Appetizer",
  "Burger",
  "Pizza",
  "Non Veg",
];
```

2. Create states for selecting and filtering data

```
  const [selectedFilter, setSelectedFilter] = useState<string[]>([]);
//check box
  const [filteredItems, setFilteredItems] = useState<Food[]>([]); //check
box
```

3. For removing any item that is not related to category/s

```
//for check box
  const HandleFilterButtonWhenClick = (selectCatagory: string) => {
    if (selectedFilter.includes(selectCatagory)) {
      const filter = selectedFilter.filter((e) => e !== selectCatagory); //remove
items that is not equal to selected check box
      setSelectedFilter(filter);
    } else {
      setSelectedFilter([...selectedFilter, selectCatagory]);
    }
  };
```

4. Use UseEffect for fetching items so we can compare it

```
useEffect(() => {
    //for check box
    const fetchData = async () => {
      const products:Food[] = await Foods15(0,14);
      console.log(products)
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
      filterItems(products);
    };

    fetchData();
  }, [selectedFilter]);
```

5. Find category and set it in single array

```
const filterItems = (products: Food[]) => {
    if (selectedFilter.length > 0) {
      const items = selectedFilter.map((selectedCatagory) => {
        const template = products.filter(
          (item) => item.category === selectedCatagory //separate and find
catagory
        );
        return template;
      });
      setFilteredItems(items.flat()); // set all catagory in one array
    } else {
      setFilteredItems([...products]);
    }
  };
```

6. Use it in html and now ready for use

```
<div className="grid lg:grid-cols-3  lg:gap-[32px] grid-col-1 gap-[16px]">
                {filteredItems.map((food) => (
                  <div
                    key={food._id}
                    className="w-auto flex flex-col items-start justify-between
gap-[8px]"
                  >
                    <div>
                      <img
                        src={food.imageUrl}
                        width={312}
                        height={267}
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
                      alt={food.name}
                    />
                  </div>
                  <div className="w-full flex flex-col items-start justify-
between gap-[8px]">
                    <h3 className="font-bold text-[18px]">{food.name}</h3>
                    <div className="w-auto flex flex-row items-start justify-
between gap-[16px]">
                      <div className="text-[#FF9F0D]">
                        ${food.originalPrice.toFixed(2)}
                      </div>
                      <div>
                        {food.price ? (
                          <div className="text-[#828282] line-through">
                            ${food.price.toFixed(2)}
                          </div>
                        ) : (
                          <div className="hidden"></div>
                        )}
                      </div>
                    </div>
                  </div>
                  <div className="flex flex-row w-full justify-between items-
center">
                    <Button
                      variant="primary"
                      size="none"
                      className="py-2 px-2 hover:bg-[#ffaf37] drop-shadow-
[3px_3px_1px_rgba(189,188,187,0.5)] active:drop-shadow-none"
                    >
                      <Link
                        href={`/Shop/${food._id}`}
                        className="text-[14px]"
                      >
                        Order Now!
                      </Link>
                    </Button>
                    <div className=" py-2 px-1 rounded-[6px] text-white w-auto
h-auto flex">
                      {food.available ? (
                        <div className="bg-[#FF9F0D] px-[6px] py-[7px] rounded-
[6px]">
                          In stock
                        </div>
                      ) : (
```

"Day 4 - Dynamic Frontend Components – Food Tuck"

```
                        <div className="bg-[#595858] px-[6px] py-[7px] rounded-
[6px]">
                          Stock Out
                        </div>
                      )}
                    </div>
                  </div>
                </div>
              ))}
            </div>
```

Button:

```
{filters.map((catagory,i) => (
                <div key={i} className="flex flex-row gap-[8px]">
                  {" "}
                  <Checkbox
                    id="terms"
                    className=""
                    onClick={() => HandleFilterButtonWhenClick(catagory)}
                  />
                  <span className="text-sm text-[#333333]">{catagory}</span>
                </div>
              ))}
```