Day 4 - Dynamic Frontend Components – Online Grocery Store

Steps Taken to Build and Integrate Components

Redux Implementation

1. Installed Redux and Redux Toolkit dependencies.
2. Configured the store with reducers to manage global state.
3. Defined slices for different data modules such as search and image handling.
4. Integrated the Redux store with the Next.js app using the `Provider` component.
5. Utilized Redux hooks (`useSelector`, `useDispatch`) to manage state across components.

Search Bar Creation

1. Designed the search bar component using React hooks.
2. Connected the search input to Redux state to track user queries.
3. Implemented a debounced function to optimize search performance.
4. Integrated API calls to fetch search results from the backend.
5. Displayed search results dynamically based on the query.

Display Image from Sanity

1. Set up Sanity CMS and connected it to the Next.js project.
2. Created image schemas in Sanity and uploaded sample images.
3. Fetched images using Sanity's client API.
4. Used Next.js Image component to optimize image loading.
5. Ensured dynamic image rendering based on content changes in Sanity.

Mobile Responsiveness

1. Utilized Tailwind CSS for responsive design.
2. Created breakpoints for various screen sizes (sm, md, lg, xl).
3. Tested responsiveness across multiple devices using Chrome DevTools.
4. Optimized component layouts to ensure a seamless experience on mobile screens.
5. Used flexbox and grid for efficient content structuring.

Challenges Faced and Solutions Implemented

Redux Setup Issues

- **Challenge:** Initial state management complexity.
- **Solution:** Used Redux Toolkit to simplify the setup with slices and middleware.

Performance Bottlenecks in Search

- **Challenge:** Frequent API calls slowing down the app.
- **Solution:** Implemented debouncing to limit API requests.

Image Rendering Delay

- **Challenge:** Slow loading of images from Sanity.
- **Solution:** Used Next.js Image optimization and caching strategies.

Mobile Layout Breakdowns

- **Challenge:** Elements misaligned on smaller screens.
- **Solution:** Applied responsive design principles using Tailwind's utility classes.

Next.js Problems

- **Challenge:** Deployment and build issues.
- **Solution:** Optimized configurations in `next.config.js` and ensured proper environment variable usage.

Best Practices Followed During Development

1. **Component-Based Architecture:** Ensured reusability and maintainability.
2. **State Management:** Centralized application state using Redux efficiently.
3. **Performance Optimization:** Implemented lazy loading and caching strategies.
4. **Code Quality:** Followed coding standards and linting rules.
5. **Testing:** Performed unit and integration testing with Jest and React Testing Library.
6. **Security:** Ensured proper handling of API keys and sensitive data.
7. **Continuous Integration/Deployment:** Used Vercel for seamless deployments.

Code Snipets

- SearchBar

```
"use client";
import { useState, useEffect } from "react";
import { client } from "@/sanity/lib/client";
import imageUrlBuilder from "@sanity/image-url";
import { ImCancelCircle } from "react-icons/im";
import "react-toastify/dist/ReactToastify.css";
import Swal from "sweetalert2";
```

```tsx
import { useDispatch, useSelector } from "react-redux";
import { addToCart } from "@/app/lib/features/todos/cartSlice";
import Header from "@/app/components/Header";
import Footer from "@/app/components/Footer";
// import Cart from "@/app/components/Cart";

export default function Productlist(props: any) {
  const [productsData, setProductsData] = useState<any[]>([]);
  const [filteredProducts, setFilteredProducts] = useState<any[]>([]);
  const [selectedProduct, setSelectedProduct] = useState<any |
null>(null);
  const [isLoading, setIsLoading] = useState(false);

  // Function to fetch data from Sanity
  const fetchSanityData = async () => {
    const [products] = await Promise.all([
      client.fetch(`
        *[_type == "product"]{
          _id,
          title,
          description,
          price,
          discountPercentage,
          isNew,
          tags,
          category->{
            _id,
            title
          },
          productImage{
            asset->{
              _id,
              url
            },
            alt
          }
        }
      `),
    ]);

    setProductsData(products);
  };

  // useEffect to fetch data when component mounts
  useEffect(() => {
```

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

 changes
  useEffect(() => {
    if (props.params.slug) {
      setFilteredProducts(
        productsData.filter((product: any) =>
          product.title.toLowerCase().includes(props.params.slug.toLower
Case())
        )
      );
    } else {
      setFilteredProducts(productsData);    }
  }, [productsData, props.params.slug]);

  const urlFor = (source: any) => {
    const builder = imageUrlBuilder(client);
    return builder.image(source);
  };

  const dispatch = useDispatch();

  const handleAddToCart = () => {
    setIsLoading(true);

    if (selectedProduct) {
      dispatch(
        addToCart({
          id: selectedProduct._id,
          name: selectedProduct.title,
          price: selectedProduct.price,
          image: selectedProduct.productImage?.asset?.url,
          quantity: quantity, // The selected quantity
        })
      );

      Swal.fire({
        position: "center",
        title: "Successfully Added",
        showConfirmButton: false,
        timer: 1500,
      }).then(() => {
        setIsLoading(false);
      });
```

```
    } else {
      Swal.fire({
        icon: "error",
        title: "Please select a product",
        showConfirmButton: false,
        timer: 1500,
      });
      setIsLoading(false);
    }
  };

  const [quantity, setQuantity] = useState(1);
  const increaseQuantity = () => setQuantity((prev) => prev + 1);
  const decreaseQuantity = () =>
    setQuantity((prev) => (prev > 1 ? prev - 1 : 1));

  return (
    <div>
      <main className="mx-0 md:mx-10">
        <div className="grid grid-cols-1 lg:grid-cols-3 gap-5 my-14 px-5
md:px-0">
          {filteredProducts.map((product: any) => (
            <div
              key={product._id}
              className="border w-full  md:w-auto p-4 h-full rounded-lg
shadow-md flex justify-center items-center flex-col"
            >
              {product.productImage?.asset?.url && (
                <img
                  src={urlFor(product.productImage.asset).width(400).url
()}
                  alt={product.productImage.alt || "Product Image"}
                  className="w-full max-h-[400px] object-scale-down mb-
4"
                />
              )}
              <h2 className="font-semibold text-2xl text-center">
                {product.title}
              </h2>
              <p className="text-green-600 font-bold mb-2 text-xl ">
                Rs. {product.price}
              </p>
              <button
                onClick={() => setSelectedProduct(product)}
```

```
                className="bg-green-700 p-3 font-bold hover:bg-green-800
rounded-xl text-white"
              >
                Add to cart
              </button>
              {product.discountPercentage && (
                <p className="text-red-500 text-sm">
                  {product.discountPercentage}% OFF
                </p>
              )}
            </div>
          ))}
        </div>

        {selectedProduct && (
          <div className="fixed top-0 left-0 w-full h-full flex justify-
center items-center bg-black bg-opacity-50 z-50">
            <div className="bg-white p-6 rounded-lg shadow-lg w-
[800px]">
              <div className="grid grid-cols-2 gap-5 ">
                <div>
                  <img
                    src={urlFor(selectedProduct.productImage.asset)
                      .width(500)
                      .url()}
                    alt={selectedProduct.productImage.alt || "Product
Image"}
                    className="w-[400px] h-[400px] object-contain mb-4"
                  />
                </div>
                <div className="space-y-5">
                  <div className="flex justify-end">
                    <ImCancelCircle
                      size={20}
                      className="hover:text-gray-400"
                      onClick={() => setSelectedProduct(null)}
                    />
                  </div>
                  <h2 className="font-bold text-4xl">
                    {selectedProduct.title}
                  </h2>
                  <p className="text-gray-
600">{selectedProduct.description}</p>
                  <p className="text-green-600 font-bold text-3xl">
                    Rs. {selectedProduct.price}
```

```
                  </p>
                  <div className="flex items-center space-x-4">
                    <button
                      onClick={decreaseQuantity}
                      className="bg-gray-200 px-4 py-2 rounded-lg text-
xl font-bold"
                    >
                      -
                    </button>
                    <span className="text-2xl font-
semibold">{quantity}</span>
                    <button
                      onClick={increaseQuantity}
                      className="bg-gray-200 px-4 py-2 rounded-lg text-
xl font-bold"
                    >
                      +
                    </button>
                  </div>
                </div>
              </div>

              <div
                onClick={() => setSelectedProduct(null)}
                className="flex items-center mt-4"
              >
                <button
                  onClick={handleAddToCart}
                  disabled={isLoading}
                  className={`p-3 font-bold rounded-xl w-full ${
                    isLoading
                      ? "bg-gray-400 cursor-not-allowed"
                      : "bg-green-700 hover:bg-green-800 text-white"
                  }`}
                >
                  {isLoading ? "Adding..." : "Add to Cart"}
                </button>
              </div>
            </div>
          </div>
        )}
      </main>
    </div>
  );
}
```

- ProductList

```
        {selectedProduct && (
          <div className="fixed top-0 left-0 w-full h-full flex justify-
center items-center bg-black bg-opacity-50 z-50">
            <div className="bg-white p-6 rounded-lg shadow-lg w-
[800px]">
              <div className="grid grid-cols-2 gap-5 ">
                <div>
                  <img
                    src={urlFor(selectedProduct.productImage.asset)
                      .width(500)
                      .url()}
                    alt={selectedProduct.productImage.alt || "Product
Image"}
                    className="w-[400px] h-[400px] object-contain mb-4"
                  />
                </div>
                <div className="space-y-5">
                  <div className="flex justify-end">
                    <ImCancelCircle
                      size={20}
                      className="hover:text-gray-400"
                      onClick={() => setSelectedProduct(null)}
                    />
                  </div>
                  <h2 className="font-bold text-4xl">
                    {selectedProduct.title}
                  </h2>
                  <p className="text-gray-
600">{selectedProduct.description}</p>
                  <p className="text-green-600 font-bold text-3xl">
                    Rs. {selectedProduct.price}
                  </p>
                  <div className="flex items-center space-x-4">
                    <button
                      onClick={decreaseQuantity}
                      className="bg-gray-200 px-4 py-2 rounded-lg text-
xl font-bold"
                    >
                      -
                    </button>
```

```
                    <span className="text-2xl font-
semibold">{quantity}</span>
                    <button
                      onClick={increaseQuantity}
                      className="bg-gray-200 px-4 py-2 rounded-lg text-
xl font-bold"
                    >
                      +
                    </button>
                  </div>
                </div>
              </div>
              <div
                className="flex items-center mt-4"
              >
                <button
                  onClick={handleAddToCart}
                  disabled={isLoading}
                  className={`p-3 font-bold rounded-xl w-full ${
                    isLoading
                      ? "bg-gray-400 cursor-not-allowed"
                      : "bg-green-700 hover:bg-green-800 text-white"
                  }`}
                >
                  {isLoading ? "Adding..." : "Add to Cart"}
                </button>
              </div>
            </div>
          </div>
        )}
    •
```

- Category

```
  <div className="flex flex-wrap gap-6 justify-center text-center mx-
10">
          {categoriesData.map((cat: any) => (
            <button
              key={cat._id}
              className={` w-36 h-36 md:w-72 md:h-52 cursor-pointer
font-bold bg-green-100 ${selectedCategory === cat._id ? " bg-green-500
text-white" : ""}`}
              onClick={() => setSelectedCategory(cat._id)}
            >
              <img
```

```
                src={urlFor(cat.image.asset).width(100).url()}
                alt={cat.title}
                className="w-full h-20 object-contain mb-2 "
              />
              {cat.title}
            </button>
          ))}
        </div>
```

- Cart

```
import React from "react";
import Link from "next/link";
import { HiShoppingCart } from "react-icons/hi";
import { removeFromCart } from "@/app/lib/features/todos/cartSlice";
import { RootState } from "@/app/lib/store";
import { useDispatch, useSelector } from "react-redux";
import { useState } from "react";
import { ImCancelCircle } from "react-icons/im";
const Cart = () => {
  const [isCartOpen, setIsCartOpen] = useState(false);
  const cartItems = useSelector((state: RootState) => state.cart.items);
  const toggleCartSidebar = () => {
    setIsCartOpen(!isCartOpen);
  };
  const dispatch = useDispatch();
  const handleRemoveItem = (id: string) => {
    dispatch(removeFromCart(id));
  };

  const totalPrice = cartItems.reduce(
    (acc, item) => acc + item.price * item.quantity,
    0
  );
  return (
    <div>
      {cartItems.length > 0 && (
        <button
          className="fixed bottom-6 right-6  gap-1 hover:bg-green-500
hover:text-black  bg-green-900  text-white rounded-full p-4 shadow-lg
flex items-center justify-center"
          onClick={toggleCartSidebar}
        >
          <HiShoppingCart size={25} />
```

```
          {cartItems.length}
        </button>
      )}
      {isCartOpen && (
        <div className="fixed inset-0 bg-gray-800 bg-opacity-50 z-50">
          <div className="fixed top-0 right-0 bg-white w-96 h-full
shadow-lg p-4">
            <h2 className="text-xl font-bold mb-4 text-center p-3 text-
white bg-green-700">
              My Cart
            </h2>
            <button
              className="absolute top-2 left-2 text-gray-700"
              onClick={toggleCartSidebar}
            >
              <ImCancelCircle size={30} />
            </button>
            <div className="mt-4">
              {cartItems.length === 0 ? (
                <p className="text-center text-gray-500">Your cart is
empty</p>
              ) : (
                cartItems.map((item) => (
                  <div
                    key={item.id}
                    className="flex justify-between items-center mb-4"
                  >
                    <div className="flex items-center space-x-4">
                      <div>
                        <h3 className="font-semibold">{item.name}</h3>
                        <p className="text-gray-600">Rs.
{item.price}</p>
                        <p className="text-gray-600">
                          Quantity: {item.quantity}
                        </p>
                      </div>
                    </div>
                    <div className="flex items-center space-x-4">
                      <p className="font-bold">
                        Rs. {item.price * item.quantity}
                      </p>
                      <button
                        onClick={() => handleRemoveItem(item.id)}
                        className="text-red-600 hover:text-red-800"
```

```
                    >
                      Remove
                    </button>
                  </div>
                </div>
              ))
            )}
          </div>

          {cartItems.length > 0 && (
            <div className="flex justify-between items-center mt-4">
              <h3 className="font-bold">Total Price:</h3>
              <p className="font-bold">Rs. {totalPrice}</p>
            </div>
          )}
          {cartItems.length > 0 && (
            <Link href="/checkout">
              <button
                onClick={toggleCartSidebar}
                className="bg-green-700 text-white font-bold w-full
py-3 my-3"
                >
                Checkout
              </button>
            </Link>
          )}
        </div>
      </div>
    )}
  </div>
);
};

export default Cart;
```

- Login

```
"use client";
import { useState } from "react";
import { useRouter } from "next/navigation";
import Image from "next/image";
import Header from "../components/Header";
import Footer from "../components/Footer";
```

```
const Login = () => {
  const router = useRouter();
  const [formData, setFormData] = useState({
    email: "",
    password: "",
  });
  const [error, setError] = useState("");

  const handleChange = (e: any) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
    setError(""); // Clear error on input change
  };

  const handleSubmit = (e: any) => {
    e.preventDefault();
    if (formData.email === "admin" && formData.password === "admin123")
{
      router.push("/studio");
    } else {
      setError("Invalid email or password");
    }
  };

  return (
    <div>
      <div className="flex justify-center items-center h-screen flex-
col">
        <div className="py-10 px-20 mb-10 border-2 space-y-4 rounded-
3xl">
          <div className="text-center text-5xl font-bold text-green-
700">
            Log in
          </div>
          <div>
            <Image
              src="/glogo.jpg"
              width={500}
              height={500}
              alt="Logo"
              className="mb-6 w-24 md:w-80"
            />
          </div>

          <form onSubmit={handleSubmit} className="space-y-4">
            <div className="space-y-2">
```

```
              <h1 className="font-bold text-green-800">Email:</h1>
              <input
                type="text"
                name="email"
                placeholder="email"
                value={formData.email}
                onChange={handleChange}
                className="border-2 w-full rounded-lg h-10 p-2"
              />
            </div>
            <div className="space-y-2">
              <h1 className="font-bold text-green-800">Password:</h1>
              <input
                type="password"
                name="password"
                placeholder="Password"
                value={formData.password}
                onChange={handleChange}
                className="border-2 w-full rounded-lg h-10 p-2"
              />
            </div>
            {error && <p className="text-red-600">{error}</p>}
            <button
              type="submit"
              className="bg-green-700 hover:bg-green-500 text-xl text-
white w-full rounded-md py-3"
            >
              Log in
            </button>
          </form>
        </div>
        <p>
          Create new account? <span className="text-blue-700">Sign
in</span>
        </p>
      </div>
    </div>
  );
};

export default Login;
```

- Checkout

```tsx
"use client";
import { useDispatch, useSelector } from "react-redux";
import { clearCart } from "@/app/lib/features/todos/cartSlice";
import { RootState } from "@/app/lib/store";
import { useState } from "react";
import Swal from "sweetalert2";
export default function Checkout() {
  const cartItems = useSelector((state: RootState) => state.cart.items);
  const dispatch = useDispatch();
  const [isLoading, setIsLoading] = useState(false);
  const [isLoading2, setIsLoading2] = useState(false);

  const totalPrice = cartItems.reduce(
    (acc, item) => acc + item.price * item.quantity,
    0
  );

  const handleOrder = () => {
    setIsLoading(true);

    setTimeout(() => {
      Swal.fire({
        position: "center",
        icon: "success",
        title: "Thanks for shopping",
        showConfirmButton: false,
        timer: 1500,
      }).then(() => {
        dispatch(clearCart());
        setIsLoading2(false);
        window.location.href = "/"; // Redirect to the home page
      });
    }, 2000); // Simulate a loading delay
  };

  const handleOrder2 = () => {
    setIsLoading2(true);

    setTimeout(() => {
      Swal.fire({
        position: "center",
        icon: "success",
        title: "Thanks for shopping",
        showConfirmButton: false,
        timer: 1500,
```

```
      }).then(() => {
        dispatch(clearCart());
        setIsLoading2(false);
        window.location.href = "/"; // Redirect to the home page
      });
    }, 2000); // Simulate a loading delay
  };

  return (
    <div className="min-h-screen bg-gray-100  flex justify-center">
      <div className="bg-white p-8 rounded-lg shadow-lg w-screen">
        <h1 className="text-center text-2xl font-bold bg-green-700 mb-6
py-3 text-white">
          Checkout
        </h1>
        <div className="grid sm:grid-cols-1 md:grid-cols-2 gap-5">
          <div>
            <h2 className="text-xl font-bold mb-4">Billing Details</h2>
            <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
              <input
                type="text"
                placeholder="Name"
                className="border border-gray-300 p-3 rounded-lg w-full"
              />
              <input
                type="email"
                placeholder="Email"
                className="border border-gray-300 p-3 rounded-lg w-full"
              />
              <input
                type="text"
                placeholder="Phone"
                className="border border-gray-300 p-3 rounded-lg w-full"
              />
              <input
                type="text"
                placeholder="Zip"
                className="border border-gray-300 p-3 rounded-lg w-full"
              />
              <input
                type="text"
                placeholder="Address"
                className="border border-gray-300 p-3 rounded-lg w-full
col-span-1 md:col-span-2"
              />
```

```
            </div>
          </div>

          <div>
            <div className="p-6 bg-gray-100 rounded-lg shadow-md">
              <h3 className="text-xl font-semibold border-b pb-2 text-
center">
                Total Cart ({cartItems.length})
              </h3>
              {cartItems.length === 0 ? (
                <p className="text-center text-gray-500">Your cart is
empty</p>
              ) : (
                cartItems.map((item) => (
                  <div
                    key={item.id}
                    className="flex justify-between items-center mt-4"
                  >
                    <div>
                      <h3 className="font-semibold">{item.name}</h3>
                      <p className="text-gray-600">Rs. {item.price}</p>
                      <p className="text-gray-600">Quantity:
{item.quantity}</p>
                    </div>
                    <div>
                      <p className="font-bold">
                        Rs. {item.price * item.quantity}
                      </p>
                    </div>
                  </div>
                ))
              )}
              <div className="flex justify-between mt-4 border-t pt-2
text-xl font-bold">
                <span>Total :</span>
                <span>Rs. {totalPrice}</span>
              </div>
            </div>

            <div className="mt-6 grid grid-cols-2 gap-4">
              <button
                onClick={handleOrder}
                disabled={isLoading}
                className={`py-3 px-6 rounded-lg shadow-md font-bold ${
```

```
                  isLoading ? "bg-gray-400" : "bg-yellow-400 text-blue-
900"
                }`}
              >
                {isLoading ? "Processing..." : "Pay Amount"}
              </button>
              <button
                onClick={handleOrder2}
                disabled={isLoading2}
                className={`py-3 px-6 rounded-lg shadow-md font-bold ${
                  isLoading2 ? "bg-gray-400" : "bg-blue-100 text-gray-
700"
                }`}
              >
                {isLoading2 ? "Processing..." : "Cash on Delivery"}
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  );
}
```

- Scripts js

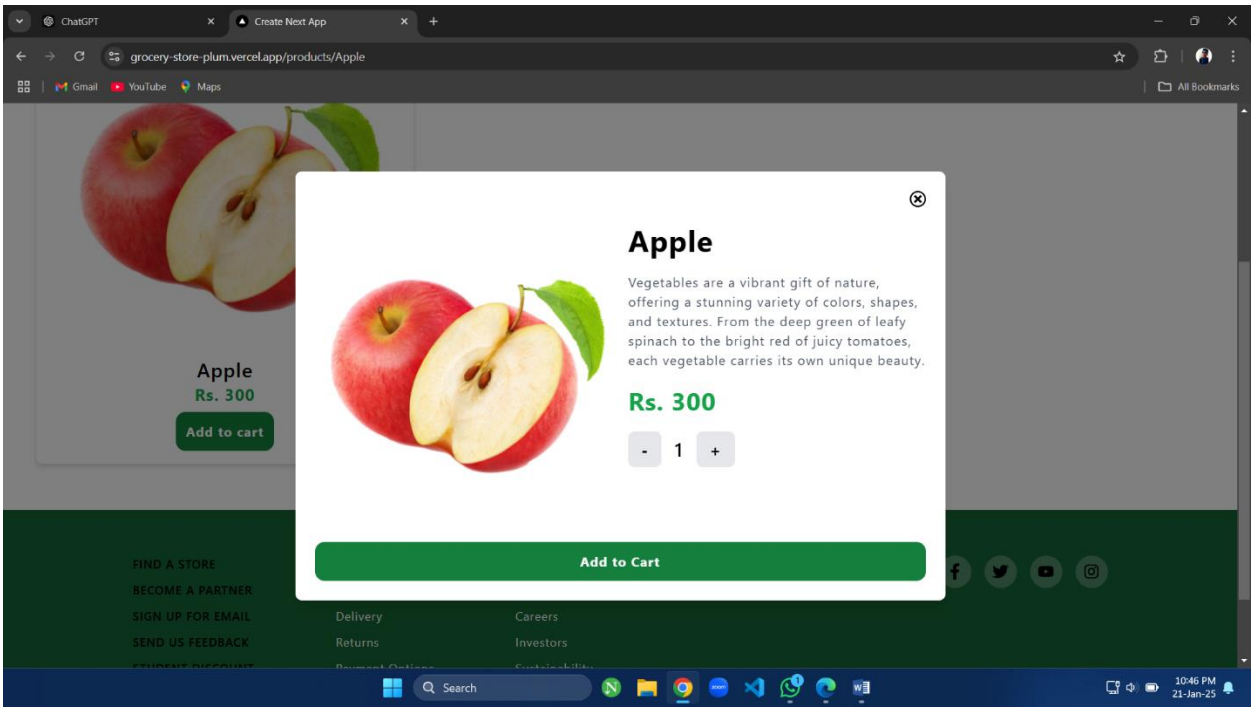
```
import { createClient } from "@sanity/client";

const client = createClient({
  projectId: "pntchuh0",
  dataset: "production",
  useCdn: true,
  apiVersion: "2025-01-13",
  token:
    "skqBbSzYvwJjKZ9DiouipHUDb5fyhEKwQ9UBrD5g48htLfUogzTZ1Gbgg0VJ27Toagx
hSZGCmUm4cFoDJsUjFcC7wIzhTcdx8A0o6Xf13bIRPALDJVDyAjlMjDSrPCHxw2No5xBy10B
3AoB1e8nNBovIblyvZxV5mj0xYjKCNCASPbjOpSnl",
});

async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
```

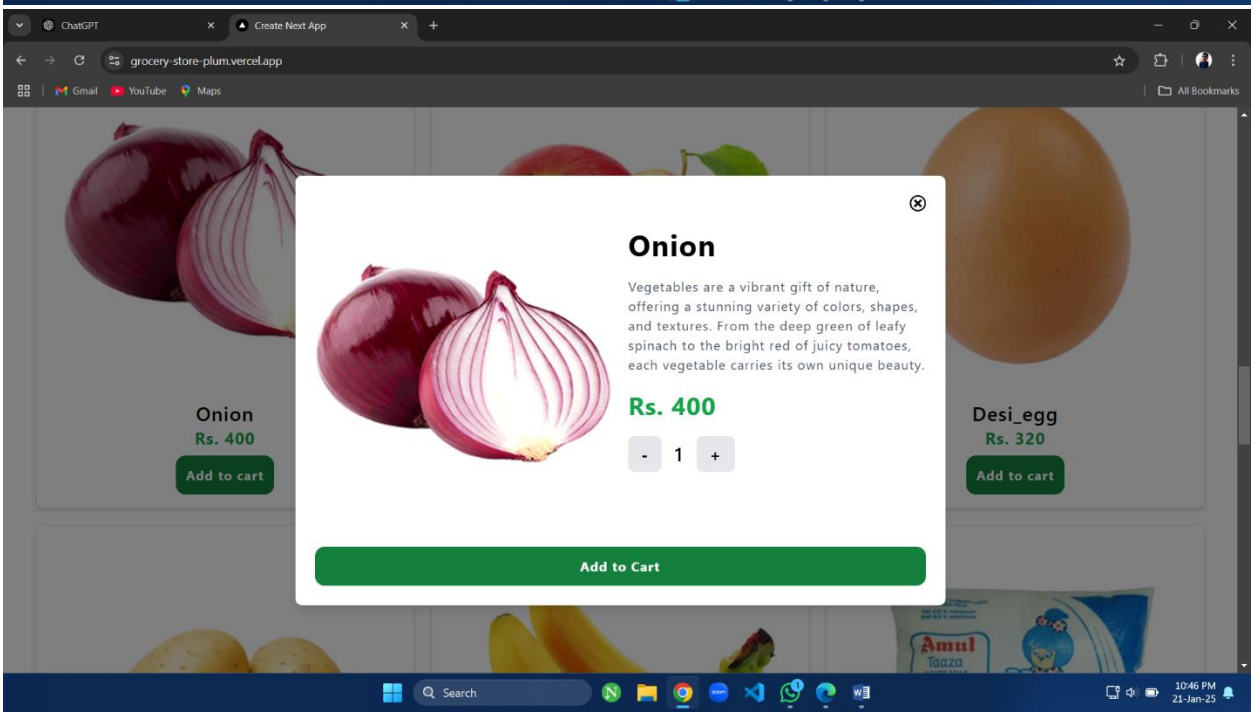
```
    const response = await fetch(imageUrl);
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload("image", bufferImage, {
      filename: imageUrl.split("/").pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error("Failed to upload image:", imageUrl, error);
    return null;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: "product",
        title: product.title,
        price: product.price,
        productImage: {
          _type: "image",
          asset: {
            _ref: imageId,
          },
        },
        tags: product.tags,
        dicountPercentage: product.dicountPercentage, // Typo in field
name: dicountPercentage -> discountPercentage
        description: product.description,
        isNew: product.isNew,
      };

      const createdProduct = await client.create(document);
      console.log(
        `Product ${product.title} uploaded successfully:`,
```

```
        createdProduct
      );
    } else {
      console.log(
        `Product ${product.title} skipped due to image upload failure.`
      );
    }
  } catch (error) {
    console.error("Error uploading product:", error);
  }
}

async function importProducts() {
  try {
    const response = await fetch(
      "https://template6-six.vercel.app/api/products"
    );

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const products = await response.json();

    for (const product of products) {
      await uploadProduct(product);
    }
  } catch (error) {
    console.error("Error fetching products:", error);
  }
}

importProducts();
```

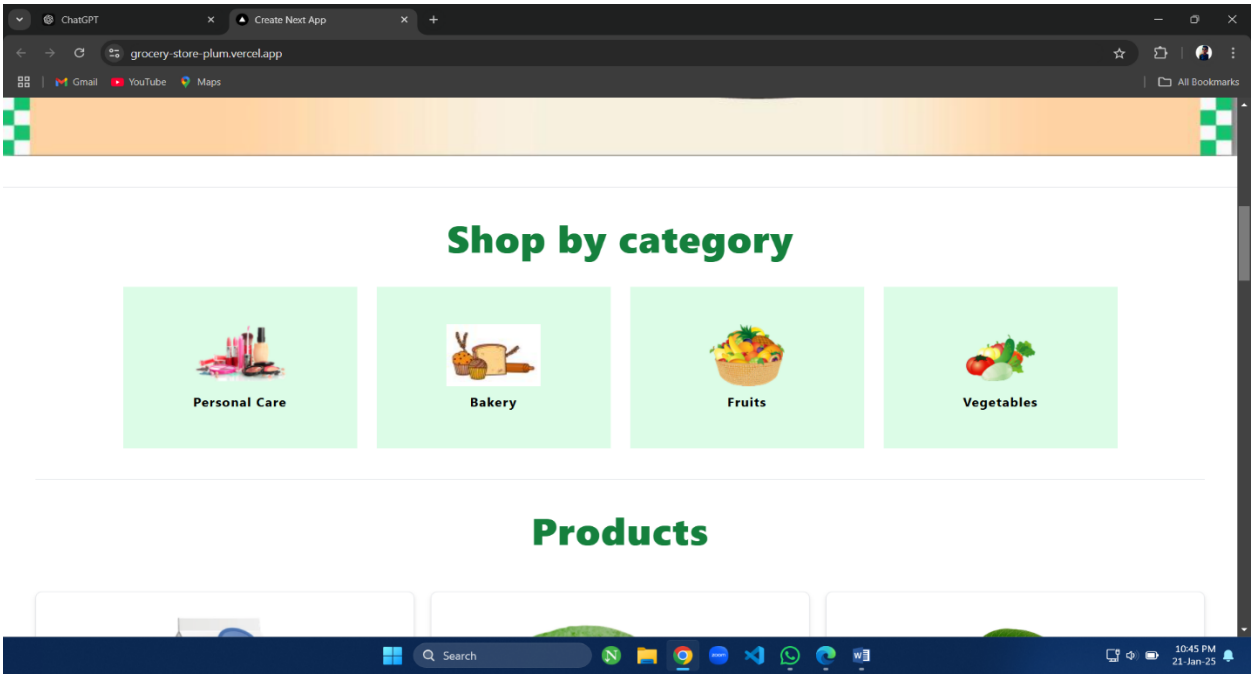
Functional Deliverables

- Individual product detail component

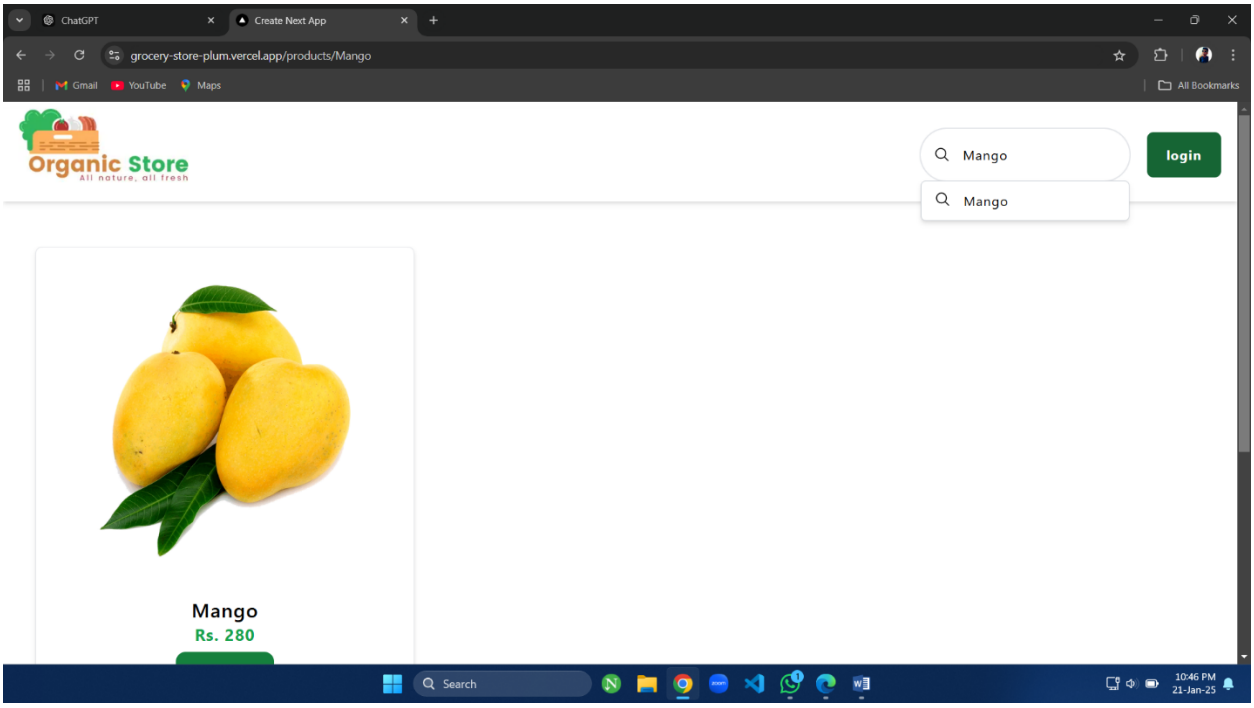


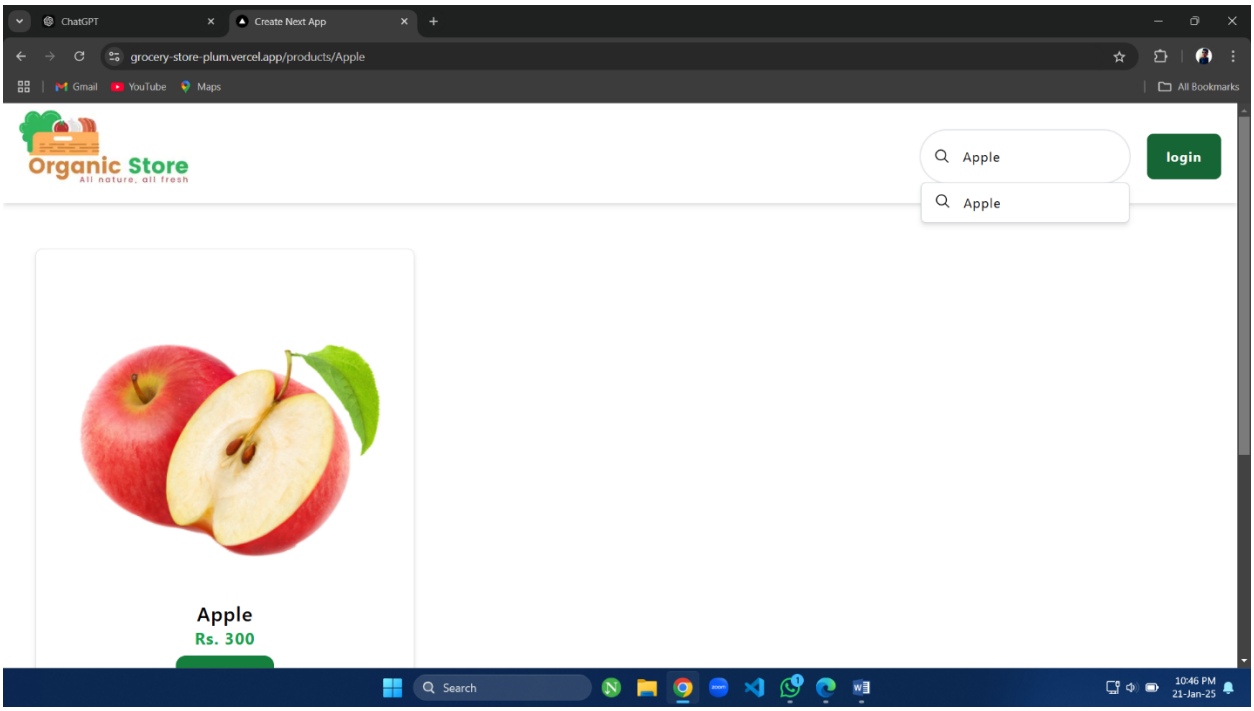


- category filters

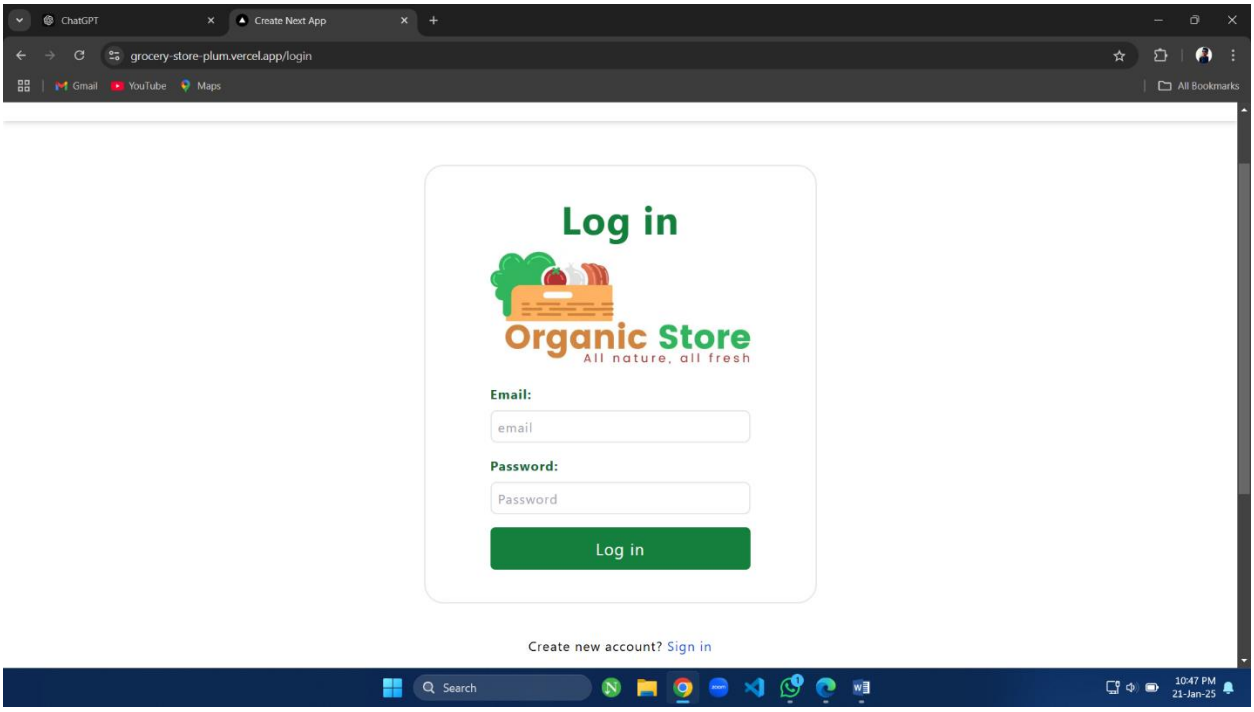

ChatGPT
Create Next App
grocery-store-plum.vercel.app
Gmail
YouTube
Maps
All Bookmarks
Shop by category
Personal Care
Bakery
Fruits
Vegetables
Products
Search
10:45 PM
21-Jan-25

- search bar

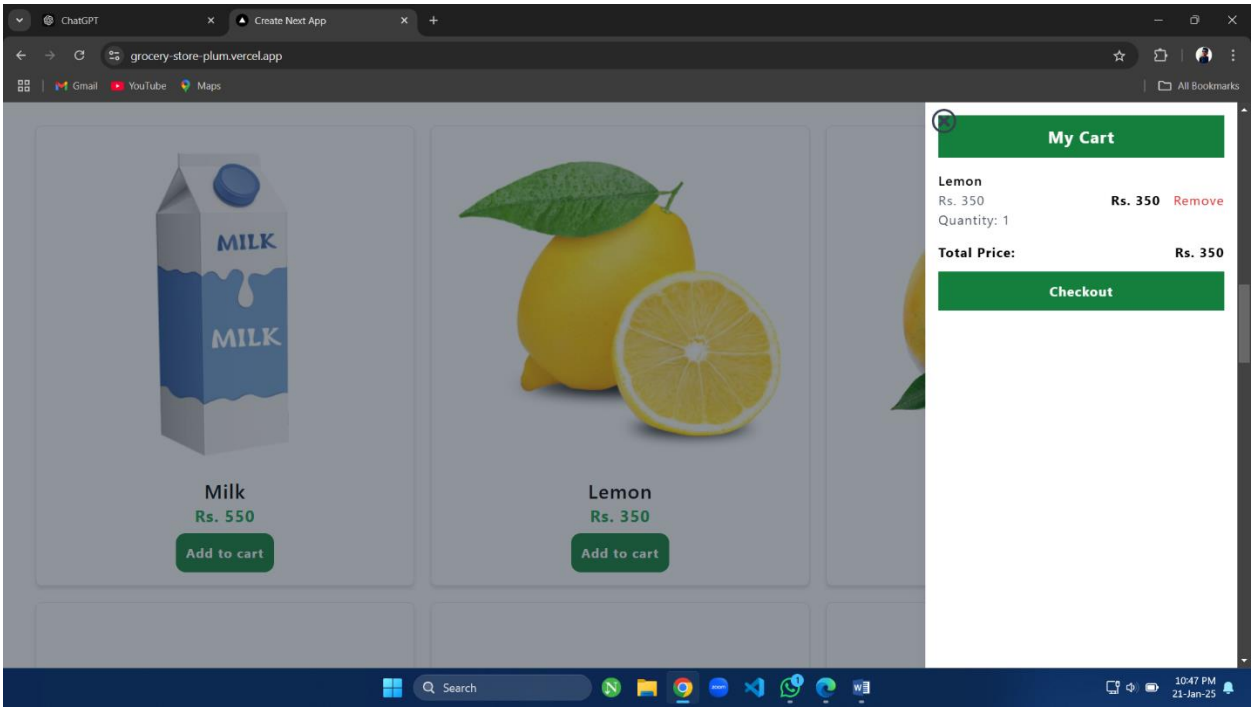

ChatGPT
Create Next App
grocery-store-plum.vercel.app/products/Mango
Gmail
YouTube
Maps
All Bookmarks
Organic Store
All nature, all fresh
Mango
login
Mango
Mango
Rs. 280
Search
10:46 PM
21-Jan-25


ChatGPT
Create Next App
grocery-store-plum.vercel.app/products/Apple
Gmail
YouTube
Maps
All Bookmarks
Organic Store
All nature, all fresh
Apple
Apple
login
Apple
Rs. 300
Search
10:46 PM
21-Jan-25

- login page


ChatGPT
Create Next App
grocery-store-plum.vercel.app/login
Gmail
YouTube
Maps
All Bookmarks
Log in
Organic Store
All nature, all fresh
Email:
email
Password:
Password
Log in
Create new account? Sign in
Search
10:47 PM
21-Jan-25

- Cart



- Checkout