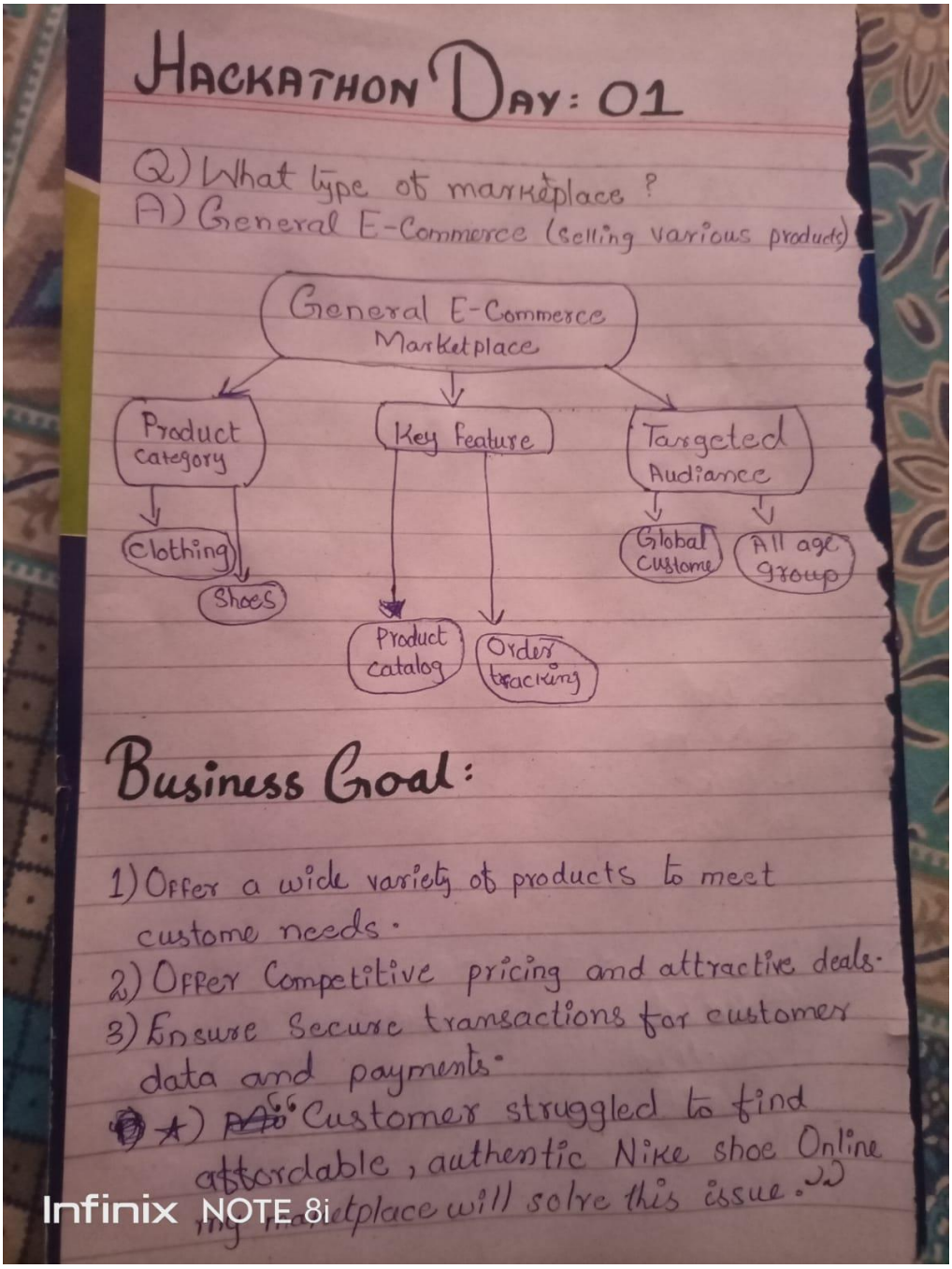
Day-1 of Hackathon:



Data Schema:

User:

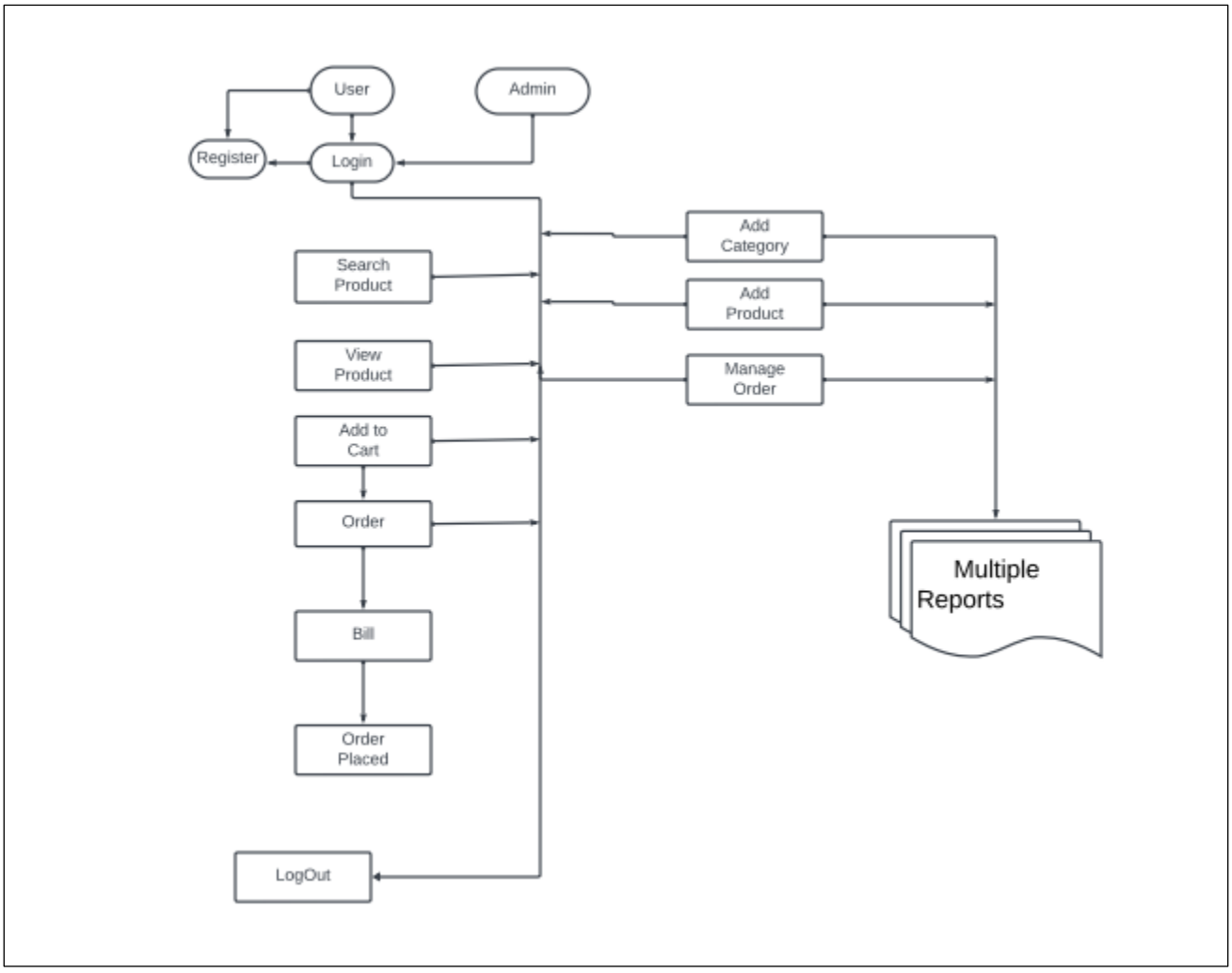
- Id
- Name
- Age
- Email
- Password

Product:

- Id
- Name
- Price
- Stock
- ⋮

Order ----> Customer
Order Id — Customer Name
Product Id — Contact info
Quantity

Day-2 of Hackathon:



Day-3 of Hackathon:

- In scripts directory this is the file of" importSanityData.mjs" for the migration of data in to sanity

```
import { createClient } from '@sanity/client';
import axios from 'axios';
import dotenv from 'dotenv';
import { fileURLToPath } from 'url';
import path from 'path';

// Load environment variables from .env.local
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);
dotenv.config({ path: path.resolve(__dirname, '../.env.local') });

// Create Sanity client
const client = createClient({
  projectId: process.env.NEXT_PUBLIC_SANITY_PROJECT_ID,
  dataset: process.env.NEXT_PUBLIC_SANITY_DATASET,
  useCdn: false,
  token: process.env.SANITY_API_TOKEN,
  apiVersion: '2021-08-31'
});
```

```
async function uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      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 uploadImageToSanity(imageUrl) {
  try {
    console.log(`Uploading image: ${imageUrl}`);
    const response = await axios.get(imageUrl, { responseType: 'arraybuffer' });
    const buffer = Buffer.from(response.data);
    const asset = await client.assets.upload('image', buffer, {
      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 importData() {
  try {
    console.log('migrating data please wait...');

    // API endpoint containing car data
    const response = await axios.get('https://template-03-
api.vercel.app/api/products');
    const products = response.data.data;
    console.log("products ==>> ", products);
```

```
    for (const product of products) {
      let imageRef = null;
      if (product.image) {
        imageRef = await uploadImageToSanity(product.image);
      }

      const sanityProduct = {
        _type: 'product',
        productName: product.productName,
        category: product.category,
        price: product.price,
        inventory: product.inventory,
        colors: product.colors || [], // Optional, as per your schema
        status: product.status,
```

```
      await client.create(sanityProduct);
    }

    console.log('Data migrated successfully!');
  } catch (error) {
    console.error('Error in migrating data ==>> ', error);
  }
}

importData();
```

- Products-Schema of the data
- file Location => sanity/schemaTypes/ products.ts

```
export const productSchema = {
  name: 'product',
  title: 'Product',
  type: 'document',
  fields: [
    {
      name: 'productName',
      title: 'Product Name',
      type: 'string',
    },
    {
      name:"slug",
      type:"slug",
      title:"Slug",
      options:{
        source:"productName"
      }
    },
    {
      name: 'category',
      title: 'Category',
      type: 'string',
    },
    {
      name: 'price',
      title: 'Price',
      type: 'number',
    },
    {
      name: 'inventory',
      title: 'Inventory',
      type: 'number',
```

```
    },
    {
      name: 'colors',
      title: 'Colors',
      type: 'array',
      of: [{ type: 'string' }],
    },
    {
      name: 'status',
      title: 'Status',
      type: 'string',
    },
    {
      name: 'image',
      title: 'Image',
      type: 'image', // Using Sanity's image type for image field
      options: {
        hotspot: true,
      },
    },
    {
      name: 'description',
      title: 'Description',
      type: 'text',
    },
  ],
}
```

- Queries for fetching data
- file Location => sanity/lib/queries.ts

```
import { groq } from "next-sanity";

//query for fetching all products from sanity
export const allProducts = groq`*[_type == "product"]`

//query for only four products fetching from sanity
export const fourProducts = groq`*[_type == "product"][0..3]`


//query for only product's categories fetching from sanity
export const productsCategory = groq`*[_type == "product"].category`
```

- fetching Data
- file Location => app/homepage/bestProducts-section.tsx

```tsx
"use client";

import { FaChevronLeft, FaChevronRight } from "react-icons/fa"; // Importing icons for navigation
import Image from "next/image"; // Importing Next.js optimized Image component
import Link from "next/link";
import { useEffect, useState } from "react";
import { Products } from "../../../types/products";
import { client } from "@/sanity/lib/client";
import { fourProducts } from "@/sanity/lib/queries";
import { urlFor } from "@/sanity/lib/image";

export default function BestProducts() {
  const [product, setProduct] = useState<Products[]>([]);

  useEffect(() => {
    async function fetchProducts() {
      const fetchedProduct: Products[] = await client.fetch(fourProducts);
      setProduct(fetchedProduct);
    }
    fetchProducts();
  }, []);
```

- Displaying of fetched data

```tsx
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4    gap-6">
        {product.map((product) => (
         <div
          key={product._id}
          className="bg-white shadow rounded-lg p-4 hover:shadow-md transition-all"
         >
          {/* Product Image */}
          <div className="w-full h-48 mb-4 flex items-center justify-center">
           {product.image && (
            <Image
             src={urlFor(product.image).url()}
             alt={product.productName}
             width={200}
             height={200}
             className="object-contain"
             priority
            />
           )}
          </div>

          {/* Product Details */}
          <div className="text-center">
```

```
        <p className="text-lg font-semibold">{product.productName}</p>
        <p className="text-gray-700 text-sm">Rs {product.price}</p>
        <p className="text-gray-500 text-xs">{product.category}</p>
      </div>
    </div>
  ))}
```

- Fetching all-products and their categories
- file Location => app/ALL-PRODUCTS/ page.tsx

```tsx
"use client"
import Image from 'next/image';
import Link from 'next/link';
import { useEffect, useState } from 'react';
import { Products } from '../../../types/products';
import { client } from '@/sanity/lib/client';
import { allProducts, productsCategory } from '@/sanity/lib/queries';
import { urlFor } from '@/sanity/lib/image';

export default function AllProducts() {

  const[product,setProduct] = useState<Products[]>([])
  const[category,setcategory] = useState<string[]>([])

  useEffect(()=>{
  async function fetchingAllProducts(){
    const fetchedProducts:Products[] = await client.fetch(allProducts)
    const fetchedCategories:string[] = await client.fetch(productsCategory)
    setProduct(fetchedProducts)
    setcategory(fetchedCategories)
  }
  fetchingAllProducts()
},[]);
```

- Displaying All-Products
- file Location => app/ALL-PRODUCTS/ page.tsx

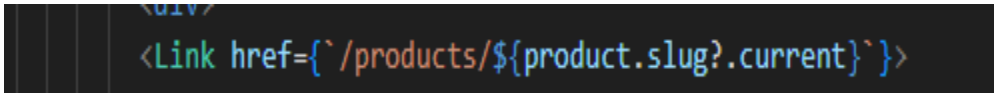
```
<h2 className="text-2xl font-bold mb-6">All Products</h2>
```

```
<div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
  {product.map((product) => (
    <div key={product._id} className="space-y-3">
      <div>
      <Link href={`/products/${product.slug?.current}`}>
      {/* Product Image */}
      <div className="w-full h-48 flex justify-center items-center bg-gray-100 rounded-lg">
        {product.image &&(
        <Image
          src={urlFor(product.image).url()}
          alt={product.productName}
          width={200}
          height={200}
          className="object-contain"
        />)}
      </div>

      {/* Product Details */}
      <div className="text-start space-y-1 mt-1">
        <h4 className="text-sm md:text-base font-medium leading-tight">
          {product.productName}
        </h4>
        <p className="text-gray-500 text-xs md:text-sm">{product.color}</p>
        <p className="text-gray-500 text-xs md:text-sm">{product.category}</p>
        <p className="text-lg font-bold">Rs {product.price}</p>
      </div>
      </Link>
      </div>
    </div>
  ))}
</div>
```

Day-4 of Hackathon:

- Making dynamic UI
- Generating slug

- file Location => app/ALL-PRODUCTS/ page.tsx



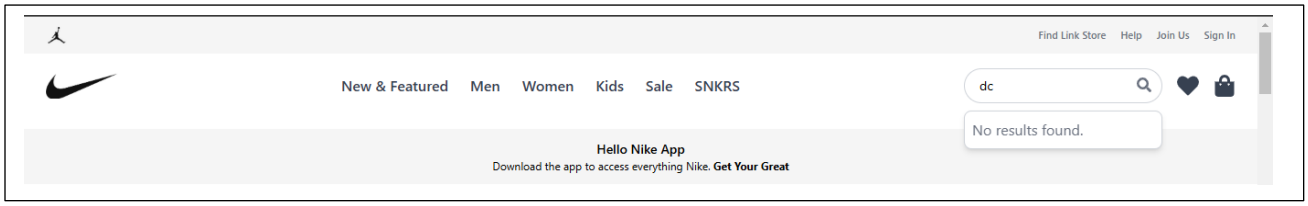
```
<Link href={`/products/${product.slug?.current}`}>
```

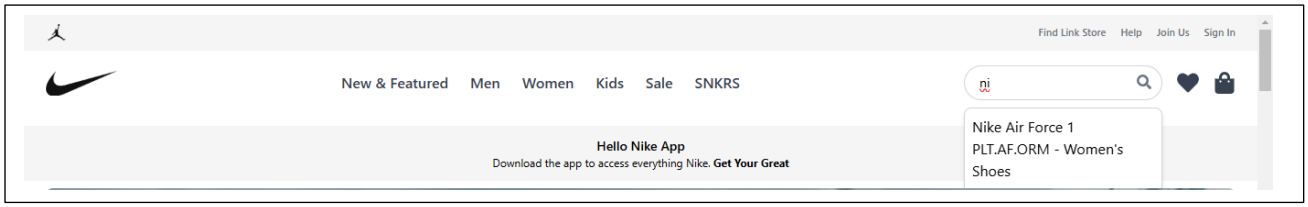
- file Location => app/products/[slug]/ page.tsx

```
import { client } from "@/sanity/lib/client";
import { Products } from "../../../../types/products";
import { groq } from "next-sanity";
import { urlFor } from "@/sanity/lib/image";
import Link from "next/link";

interface productPageProps{
  params:Promise<{slug:string}>
}

async function getProduct(slug:string):Promise<Products> {
  return client.fetch(
    groq`*[_type=="product" && slug.current==$slug][0]{
    _id,
    productName,
    image,
    type,
    description,
    price,
    }`,{slug}
  )
}

export default async function ProductPage({params}:productPageProps){
  const {slug} = await params
  const product = await getProduct(slug)


  return (
    <div className="flex flex-col lg:flex-row items-center lg:items-start justify-center px-4 sm:px-6 lg:px-12
py-8">
      {/* Product Image */}
      <div className="w-full max-w-sm">
       {product.image && (
         <img
           src={urlFor(product.image).url()}
```

```
          alt={product.productName}
          className="w-full h-auto object-contain"
        />
       )}
      </div>

      {/* Product Details */}
      <div className="mt-6 lg:mt-0 lg:ml-10 text-center lg:text-left">
       <h1 className="text-2xl font-bold mb-4">{product.productName}</h1>
       <p className="text-gray-600 mb-6">{product.description}</p>
       <p className="text-lg font-semibold mb-6">Rs {product.price}</p>
       <Link href="/cart">
       <button className="px-6 py-3 bg-black text-white text-sm font-medium rounded-md hover:bg-gray-
800 transition-all">
         Add to Cart
       </button></Link>
      </div>
     </div>
   );
}
```

- Adding search Functionality





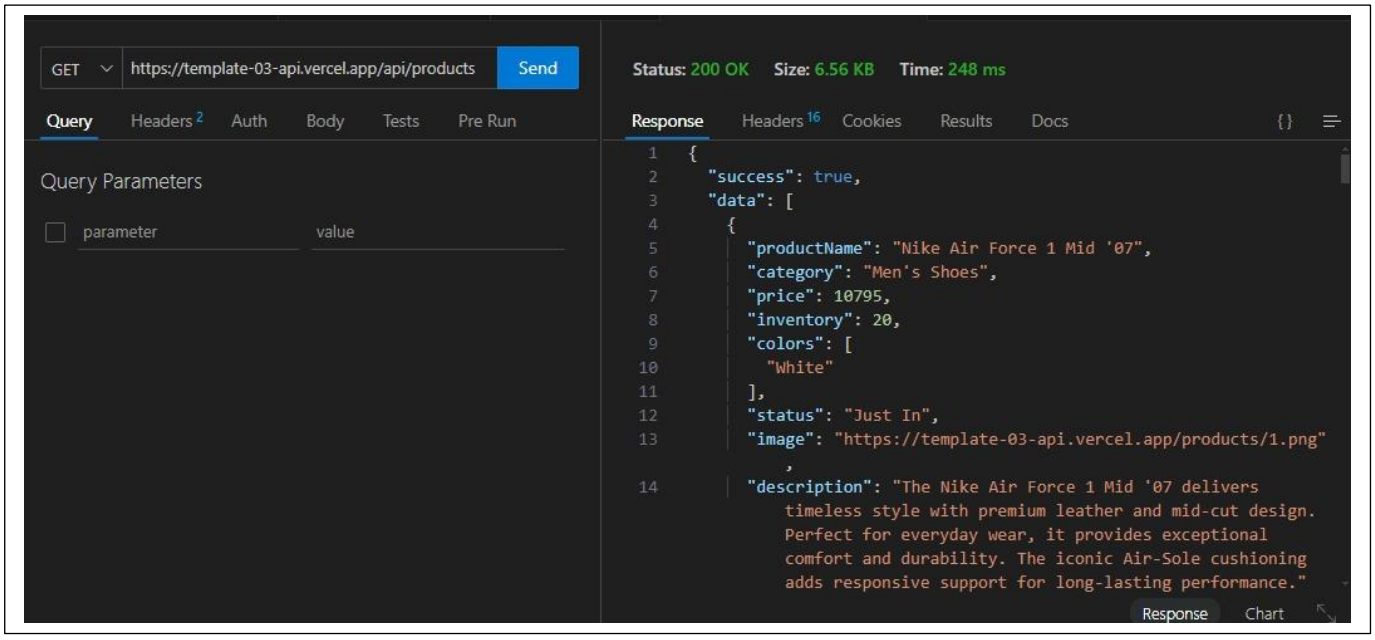
- Add to Cart functionality:
    - I couldn't implement the Add to Card functionality in my project due to time constraints. I'll try to add it later as the deadline is near. I'll make sure to implement it later, but for now, I couldn't do it.

Day-5 of Hackathon:

- Lighthouse report of my Marketplace

lightouse.pdf

- Testing API



- I'm pleased to report that I've conducted a comprehensive review of our marketplace, encompassing error handling and other critical aspects. Our tests indicate that all components are functioning as expected, with data being accurately retrieved and displayed. Furthermore, we've successfully imported data onto the

- Testing Report

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Assigned To	Remarks	
2	TC001	Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	High	-	No issues found
3	TC002	Test API error handling	Disconnect API > Refresh page	Correctly Integrated API also showing Ui through API	There is no error	Passed	Medium	-	Handled gracefully
4	TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart Functionality not working	Cart updates not expected	Not Passed	Low	-	Not working
5	TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	Medium	-	Test successful