

Name: Saba Muhammad Riaz

ID: 00485630

Template: 0

Name: Meubel House

Hackathon 3 Marketplace Builder

DAY 1: LAYING THE FOUNDATION FOR MARKETPLACE JOURNEY

Marketplace Journey Hackathon Day 01

01- My Market place Type: General E-Commerce

Because E-Commerce refers to buying and selling of goods or services over the internet. It includes transactions across website, mobile apps, and social platforms.

Primary purpose

The primary purpose of an E-Commerce is to provide a seamless and efficient online marketplace for customers to discover, purchase, and receive goods or services, while enabling business to reach a wider audience, streamline operations, and grow revenue.

My Business Goals

Launch & Build Awareness

1. Develop a fully functional e-commerce website with an intuitive UI/UX.
2. Launch a marketing campaign to attract an initial customer base.
3. Establish social media presence and drive traffic through ads and promotions.

Achieve Early Sales Milestone

1. Generate number of sales within the first [timeframe]
2. Acquire registered users or newsletters sign-ups.

Optimize Website Performance

Ensure fast loading times and mobile responsiveness.
Resolve initial bugs or glitches based on user feedback.

Data Schema

1- PRODUCT:

- Each product has a unique product ID, Name, Price, and Category.

2. INVENTORY:

- The inventory tracks the stock level and Reorder level for each product

3- Supplier:

- Supplier provide the products listed in the market place and Each Supplier is associated with multiple products

Categories:

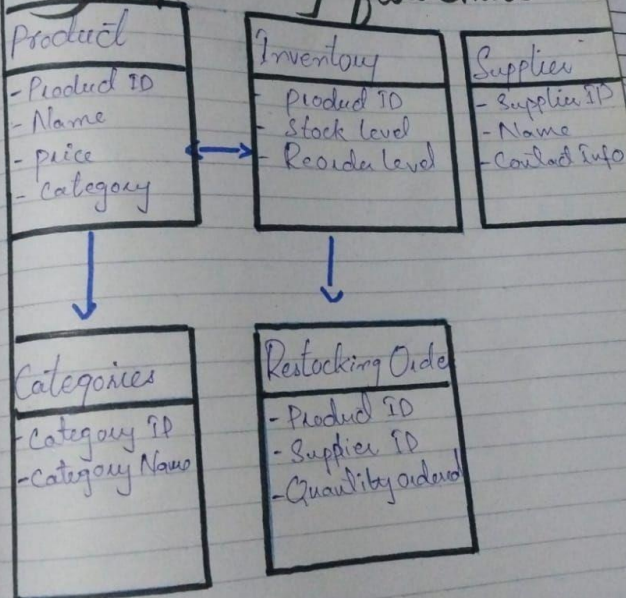
- products belong to a specific category (e.g. electronics, clothing)

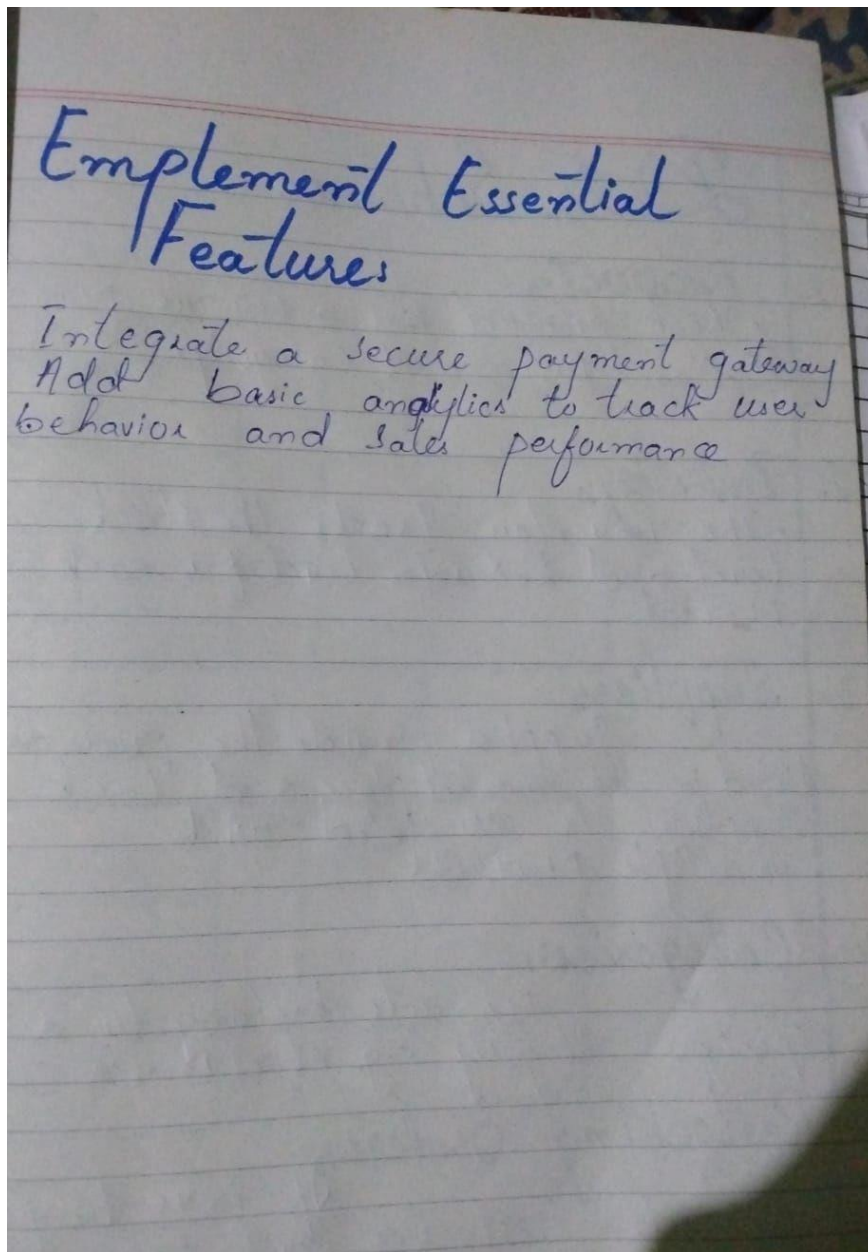
Restocking Orders:

- If inventory reaches the restocker level is

Restocking order is created request
more products from suppliers

Explain by flowchart





Hackathon 3 Marketplace Builder

Day 2 - Planning The Technical Foundation

My Business Goals

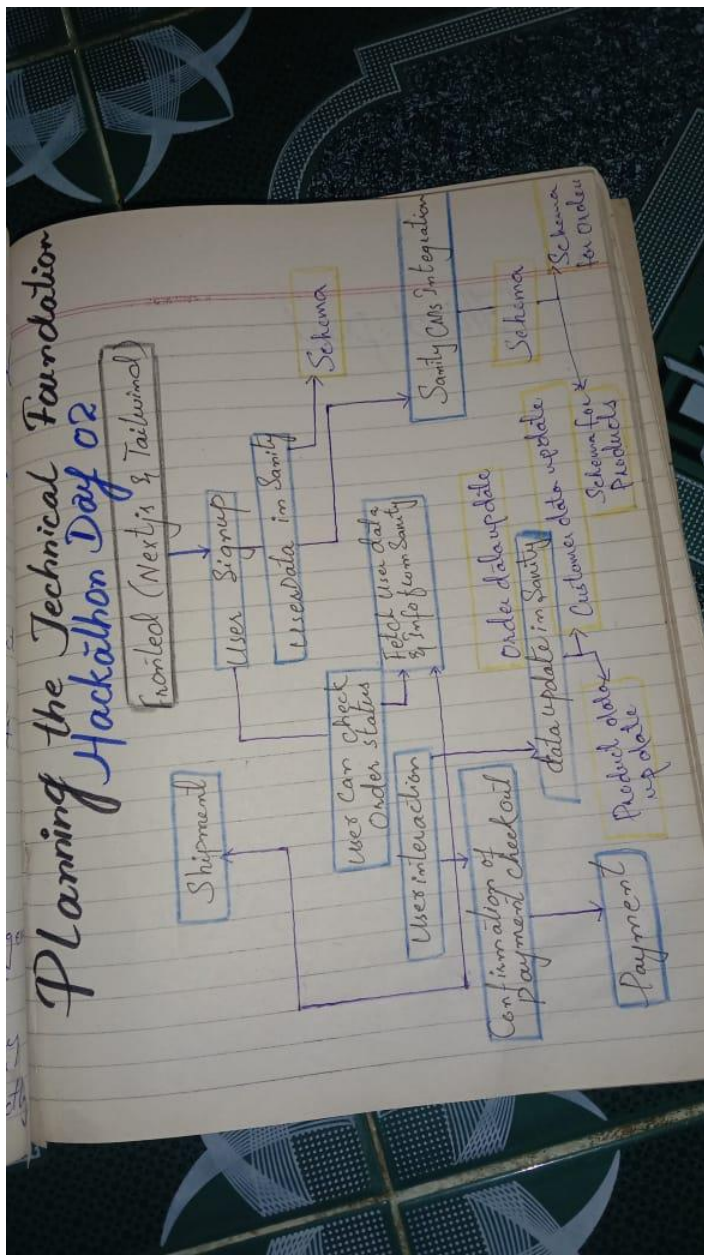
Planning the Technical Foundation

Worked (Next.js & Tailwind)

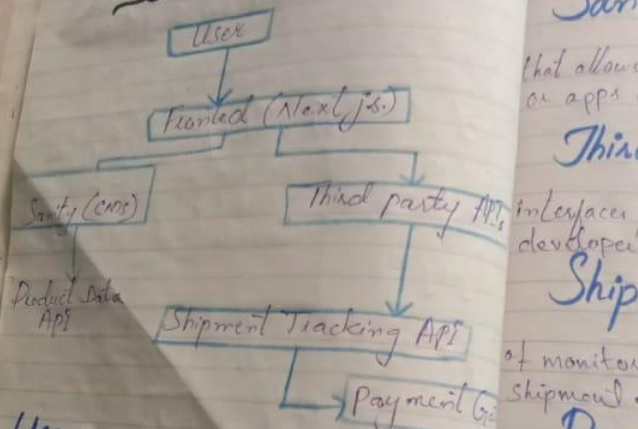
```
graph TD; Signup[User Signup] --> Data[User Data in Sanity]; Signup --> Schema1[Schema]; Data --> Fetch[Fetch User data & info from Sanity]; Fetch --> Check[User Can Check Order status]; Check --> Interact[User interaction]; Interact --> Confirm[Confirmation of Payment checkout]; Confirm --> Payment[Payment]; Payment --> Product[Product data update]; Payment --> Order[Order data update]; Payment --> Sanity[data update in Sanity]; Product --> Schema2[Schema for Products]; Order --> Schema3[Schema for Order]; Sanity --> Integration[Sanity Calls Integration]; Integration --> Schema4[Schema];
```

The diagram illustrates the technical foundation for an e-commerce application using Next.js and Tailwind. The flowchart shows the sequence of operations and data flow:

- User Signup** leads to **User Data in Sanity** and **Schema**.
- User Data in Sanity** leads to **Fetch User data & info from Sanity**.
- Fetch User data & info from Sanity** leads to **User Can Check Order status**.
- User Can Check Order status** leads to **User interaction**.
- User interaction** leads to **Confirmation of Payment checkout**.
- Confirmation of Payment checkout** leads to **Payment**.
- Payment** leads to **Product data update**, **Order data update**, and **data update in Sanity**.
- Product data update** leads to **Schema for Products**.
- Order data update** leads to **Schema for Order**.
- data update in Sanity** leads to **Sanity Calls Integration**.
- Sanity Calls Integration** leads to **Schema**.



Planning the Technical Foundation



User:

Users refer to the individuals who interact with the application, website or service.

Frontend (Next.js)

Frontend refers to the Client-Side part of a web or mobile application—everything that users interact with directly in their browser or app.

Sanity:- (CMS)

Sanity is a Headless Content management that allows you to create & manage content for website or apps and any digital platforms -

Third Party APIs:-

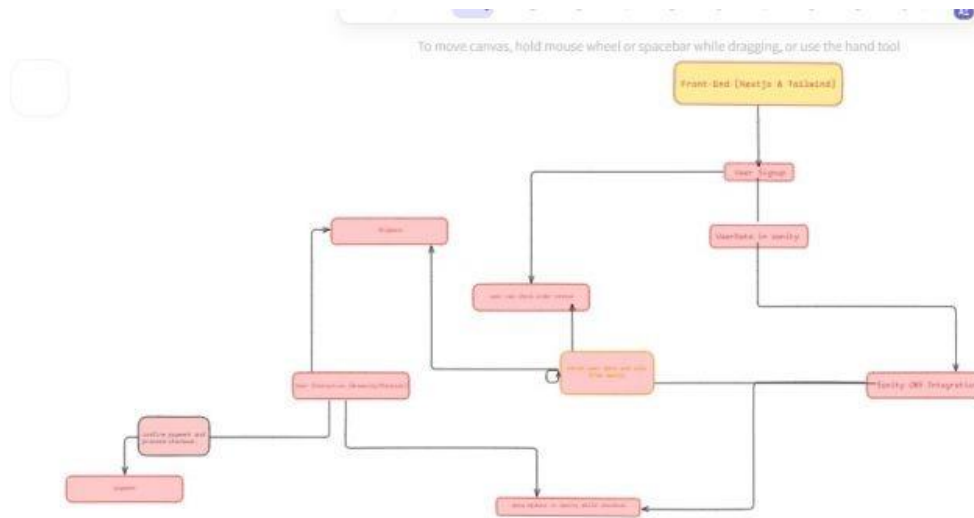
Third party APIs are interfaces provided by external companies that allows developer to access their data etc.

Shipment Tracking:-

It is a process of monitoring the movement of a package or shipment.

Payment Gateway

It is a service that facilitates online transaction between a customer and a business, acting as the intermediary between the two.



Hackathon 3 Marketplace Builder

Day 3 - API Integration and Data Migration

Integrate APIs

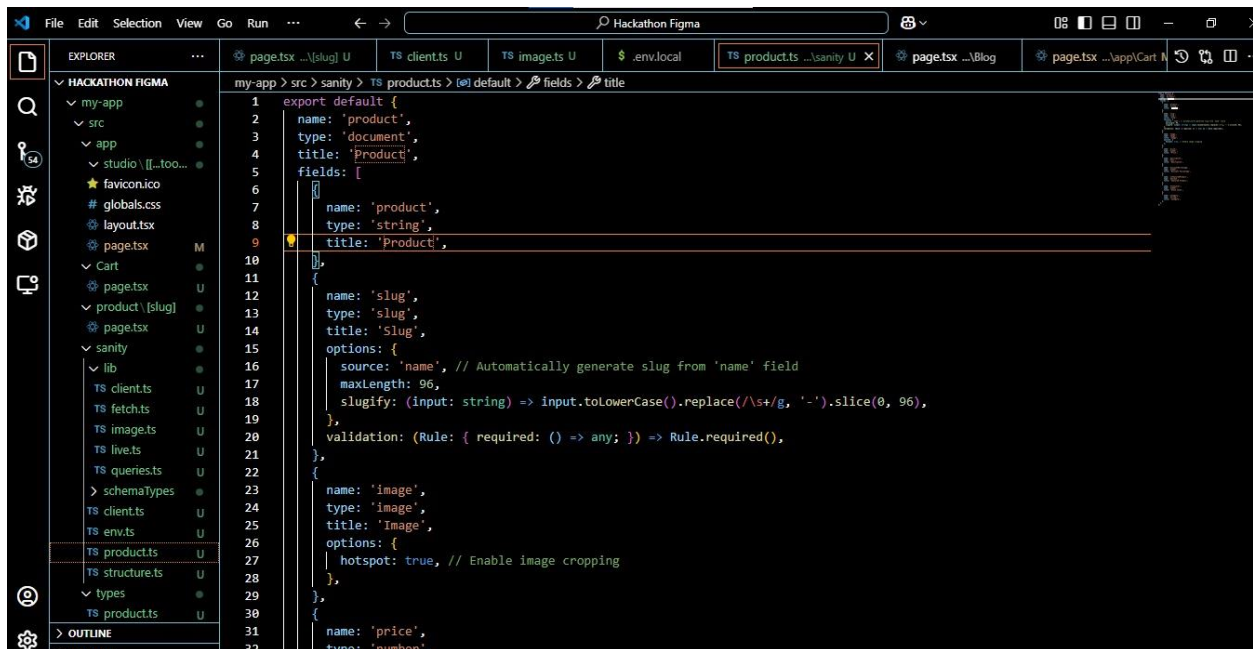
This document provides detailed of the API Integration process, Schema adjustment and Data Migration into Sanity CMS.

Integrating APIs and migrating data into Sanity CMS to build a functional marketplace backend.

Template 0

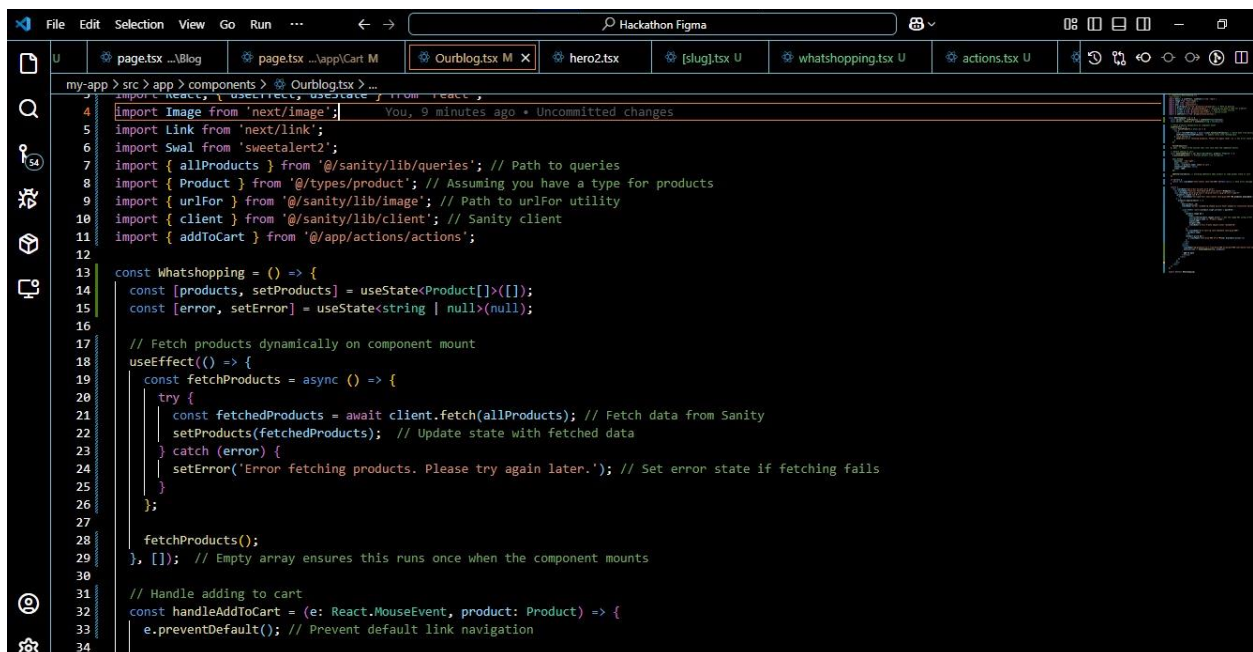
Using given APIs to populate their Sanity CMS and Import data manually.

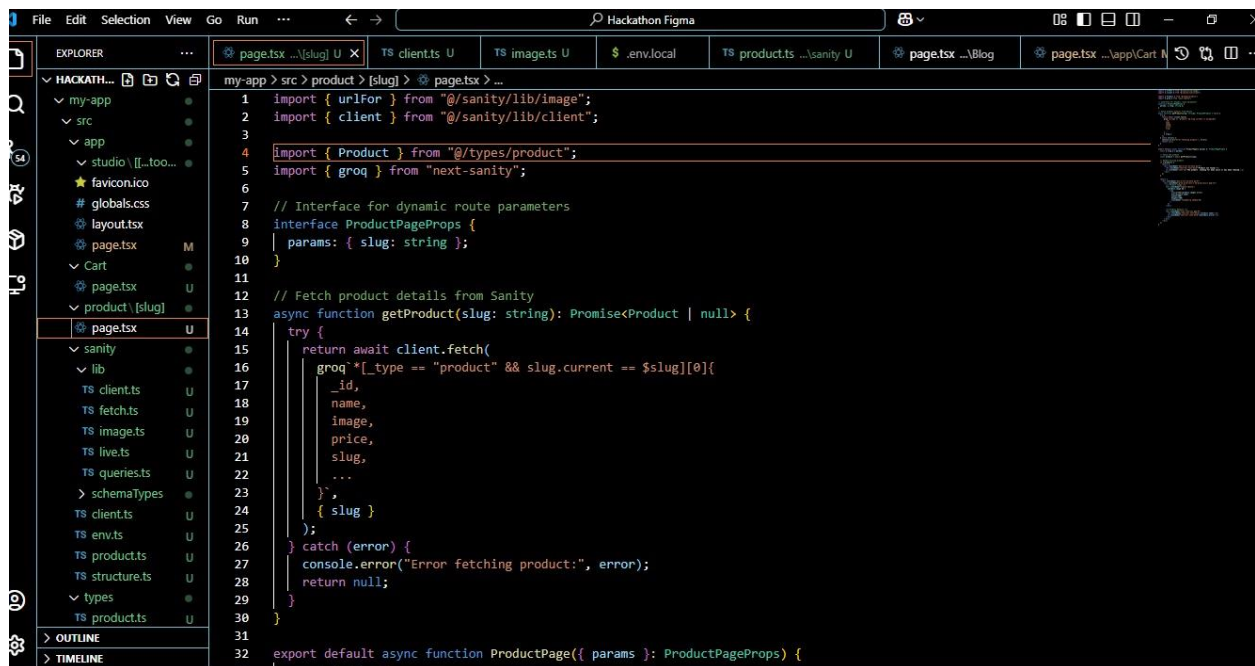
Validate and Adjust Schema



Data Fetching

Fetch and transform data from the API according to my template 0

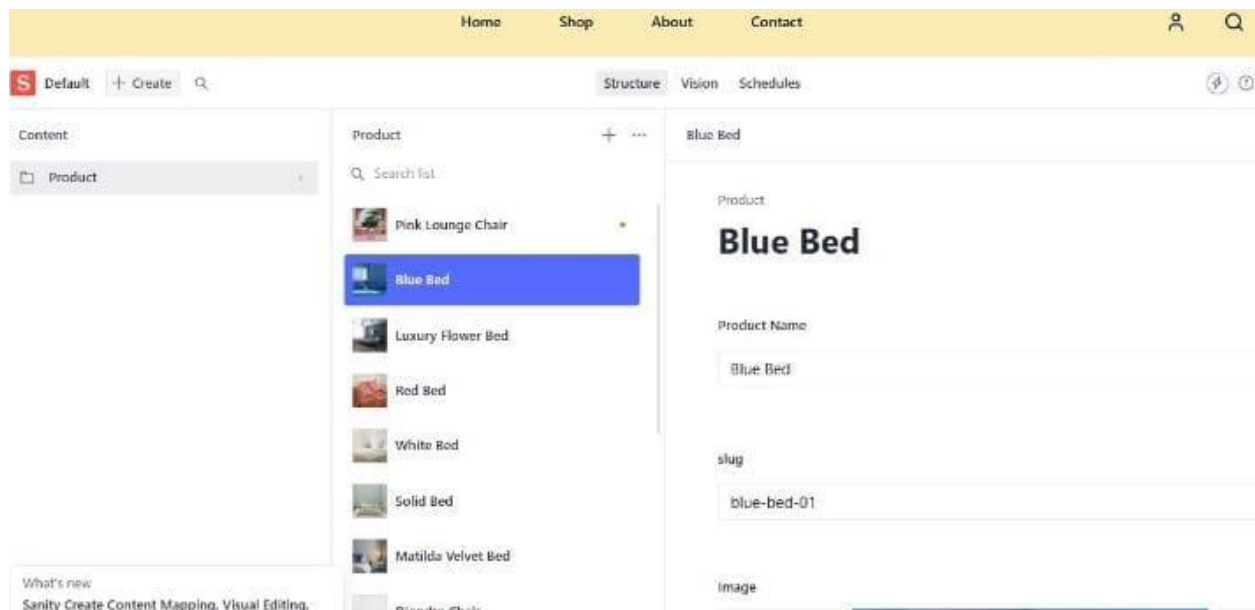




```
1 import { urlFor } from "@sanity/lib/image";
2 import { client } from "@sanity/lib/client";
3
4 import { Product } from "@types/product";
5 import { groq } from "next-sanity";
6
7 // Interface for dynamic route parameters
8 interface ProductPageProps {
9   params: { slug: string };
10 }
11
12 // Fetch product details from Sanity
13 async function getProduct(slug: string): Promise<Product | null> {
14   try {
15     return await client.fetch(
16       groq`*[_type == "product" && slug.current == $slug][0]{
17         _id,
18         name,
19         image,
20         price,
21         slug,
22         ...
23       },
24       { slug }
25     );
26   } catch (error) {
27     console.error("Error fetching product:", error);
28     return null;
29   }
30 }
31
32 export default async function ProductPage({ params }: ProductPageProps) {
```

Manual Import Data

Export data from the API by using Sanity.




Slug

armchair-chair-set

Generate


Image



Price


850

Description




Stylish Armchair
Price: \$780

Add To Cart




Hans Wegner Style Three-Legged Shell Chair
Price: \$990

Add To Cart




Nautilus Lounge Chair
Price: \$1450

Add To Cart




Diondre Chair
Price: \$720

Add To Cart




Matilda Velvet Bed
Price: \$600

Add To Cart




White Bed
Price: \$120

Add To Cart



Red Bed
Price: \$320



Hackathon 3 Marketplace Builder

Day 4 – Building Dynamic Frontend Components For Marketplace

Day 3 Recap

On Day 3, I focused on API integration and data migration to prepare my website for dynamic data handling.

Familiarized myself with the API documentation and the provided endpoints.

Tested the endpoints using Postman

Designed and implemented content models in Sanity CMS to store relevant data.

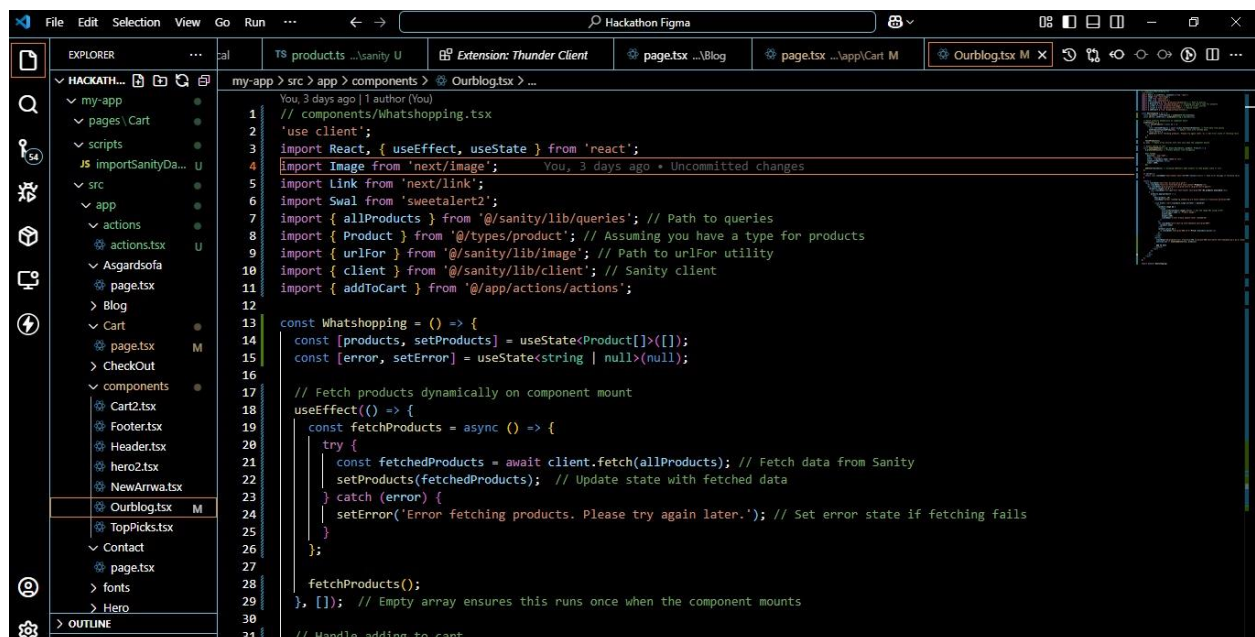
Migrated data from external sources APIs to Sanity CMS.

Implemented error handling and response parsing to ensure robustness in API integrations.

Day 4

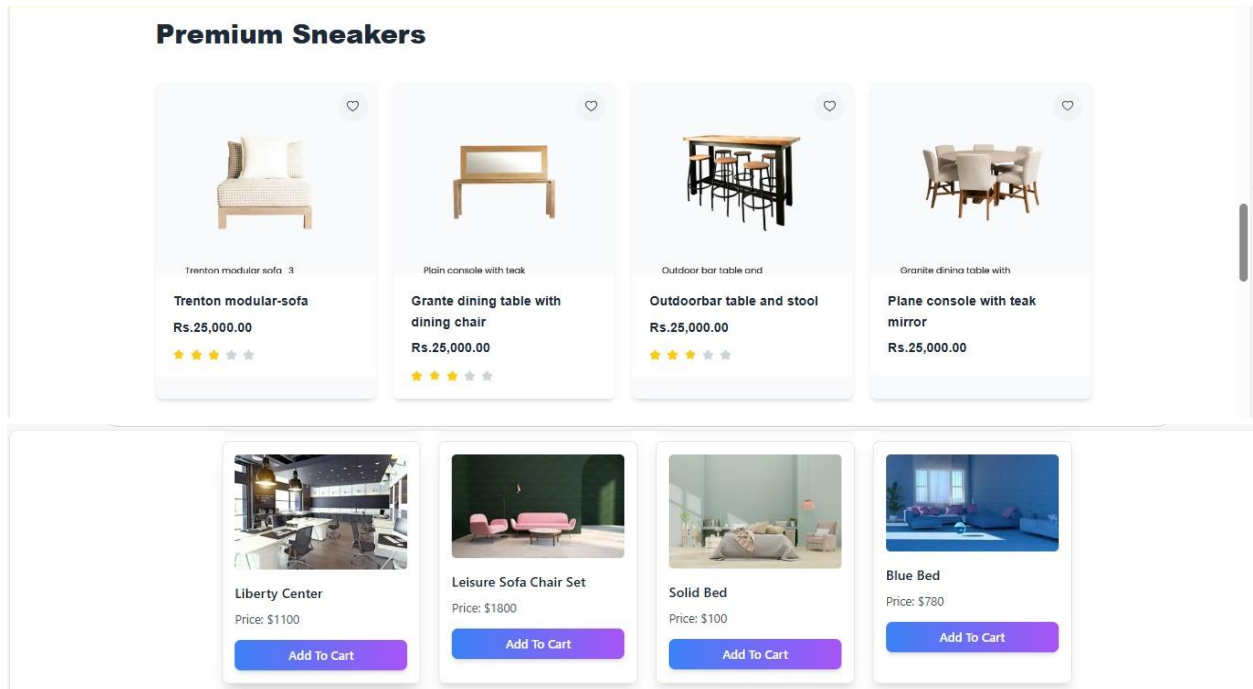
Building Dynamic Frontend Components to Display Data from Sanity CMS or APIs

- Developed dynamic frontend components that fetch and display data from Sanity CMS or external APIs.
- Used **React** for building components.
- **Sanity client** for retrieving data from Sanity



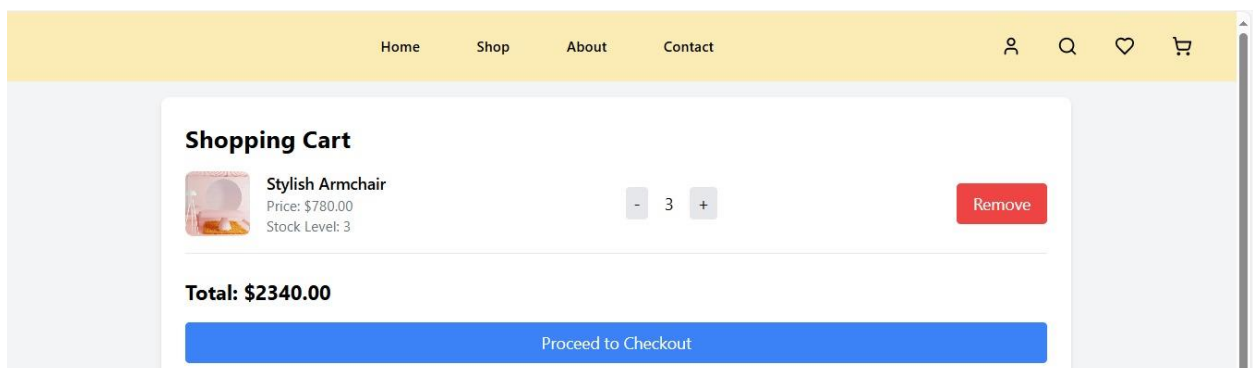
```
File Edit Selection View Go Run ... Hackathon Figma
EXPLORER
  my-app
    pages
      Cart
    scripts
    src
      app
        actions
        Asgardsofa
        page.tsx
        Blog
        Cart
        page.tsx
        Checkout
        components
          Cart2.tsx
          Footer.tsx
          Header.tsx
          hero2.tsx
          NewArrwa.tsx
          Ourblog.tsx
          TopPicks.tsx
        Contact
        page.tsx
        fonts
        Hero
    OUTLINE
    TS products.ts ...sanity U
    Extension: Thunder Client
    page.tsx ...Blog
    page.tsx ...app\Cart M
    Ourblog.tsx M X
  my-app > src > app > components > Ourblog.tsx > ...
  You, 3 days ago | 1 author (You)
  // components/whatsshopping.tsx
  1 'use client';
  2
  3 import React, { useEffect, useState } from 'react';
  4 import Image from 'next/image'; // You, 3 days ago + Uncommitted changes
  5 import Link from 'next/link';
  6 import Swal from 'sweetalert2';
  7 import { allProducts } from '@sanity/lib/queries'; // Path to queries
  8 import { Product } from '@types/product'; // Assuming you have a type for products
  9 import { urlFor } from '@sanity/lib/image'; // Path to urlFor utility
  10 import { client } from '@sanity/lib/client'; // Sanity client
  11 import { addToCart } from '@app/actions/actions';
  12
  13 const Whatshopping = () => {
  14   const [products, setProducts] = useState<Product[]>([]);
  15   const [error, setError] = useState<string | null>(null);
  16
  17   // Fetch products dynamically on component mount
  18   useEffect(() => {
  19     const fetchProducts = async () => {
  20       try {
  21         const fetchedProducts = await client.fetch(allProducts); // Fetch data from Sanity
  22         setProducts(fetchedProducts); // Update state with fetched data
  23       } catch (error) {
  24         setError('Error fetching products. Please try again later.');// Set error state if fetching fails
  25       }
  26     };
  27
  28     fetchProducts();
  29   }, []); // Empty array ensures this runs once when the component mounts
  30
  31   // Handle adding to cart
```

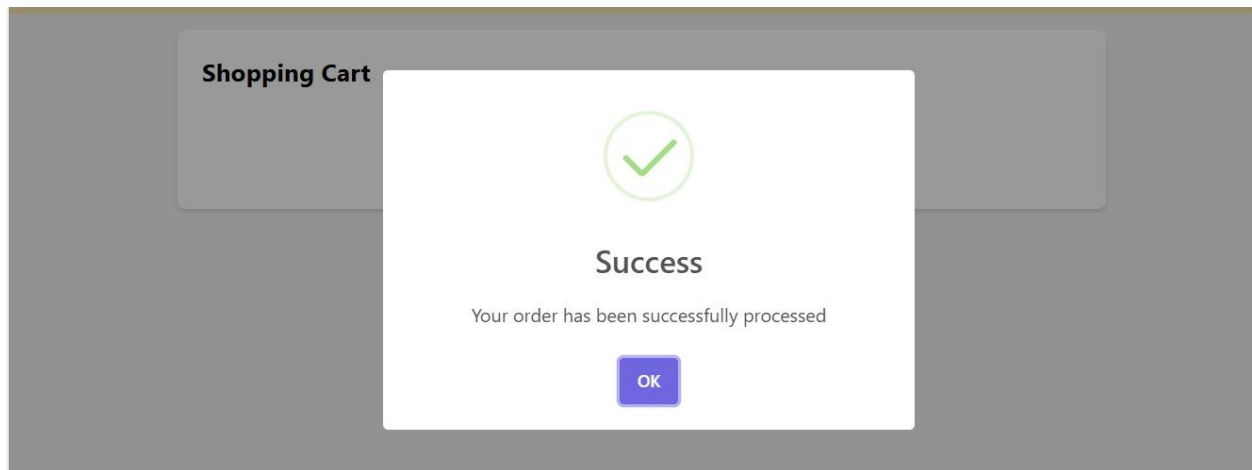
Frontend components to display data from Sanity CMS



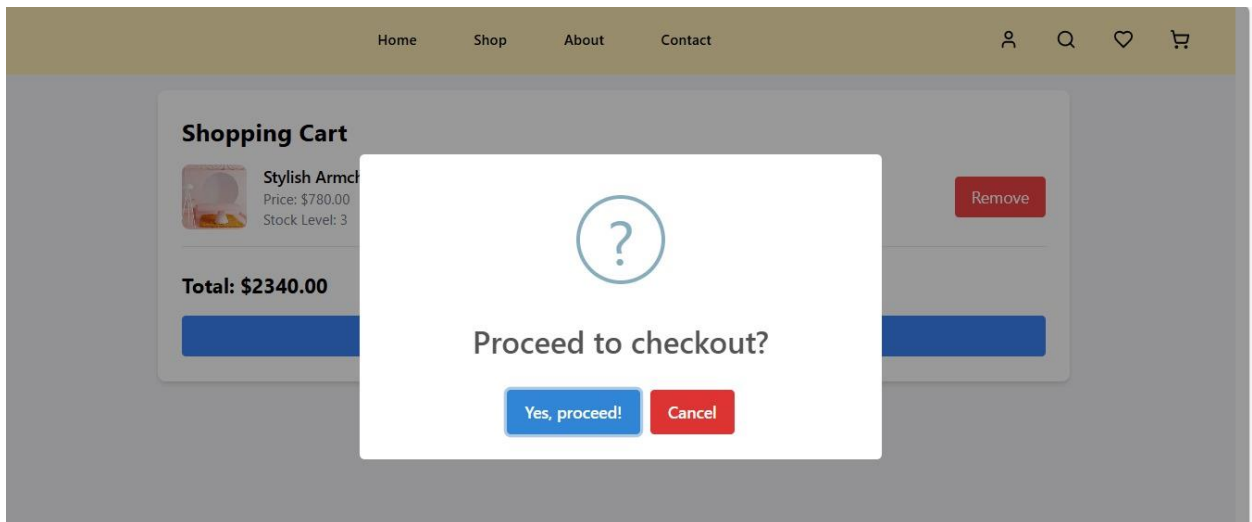
Cart Component

Display added items, quantity, and total price.






Checkout Component



SignUp Page

First Name	<input type="text" value="Saba"/>	Last Name	<input type="text" value="Riaz"/>
Email Id	<input type="text" value="sabaranicool94@gmail.com"/>	Mobile No.	<input type="text" value="03402153070"/>
Password	<input type="password" value="....."/>	Confirm Password	<input type="password" value="....."/>



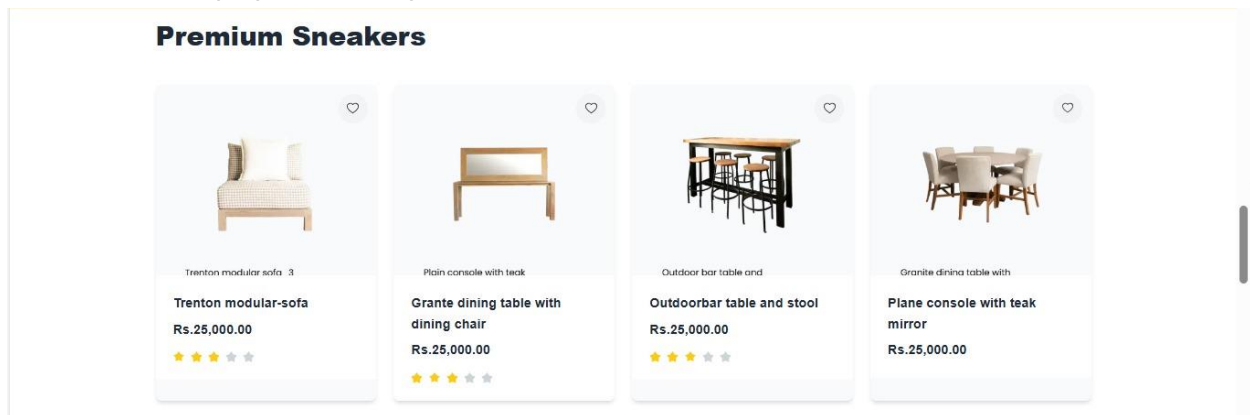
Day 5 - Testing, Error Handling, and Backend Integration Refinement

Functional Testing

All marketplace features have been thoroughly tested and validated to ensure they are functioning as intended.

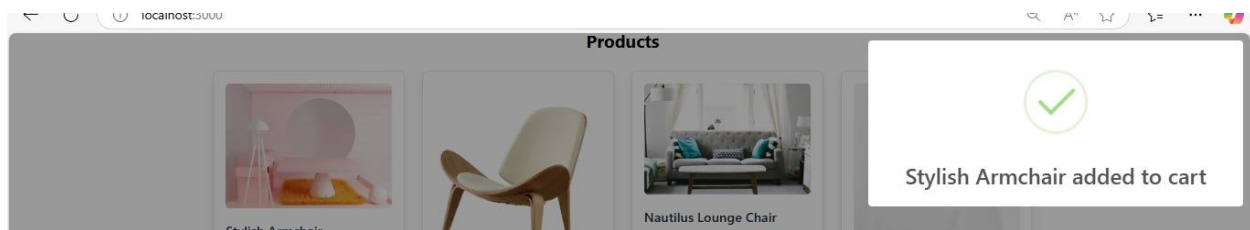
Product listing

Products are displayed correctly



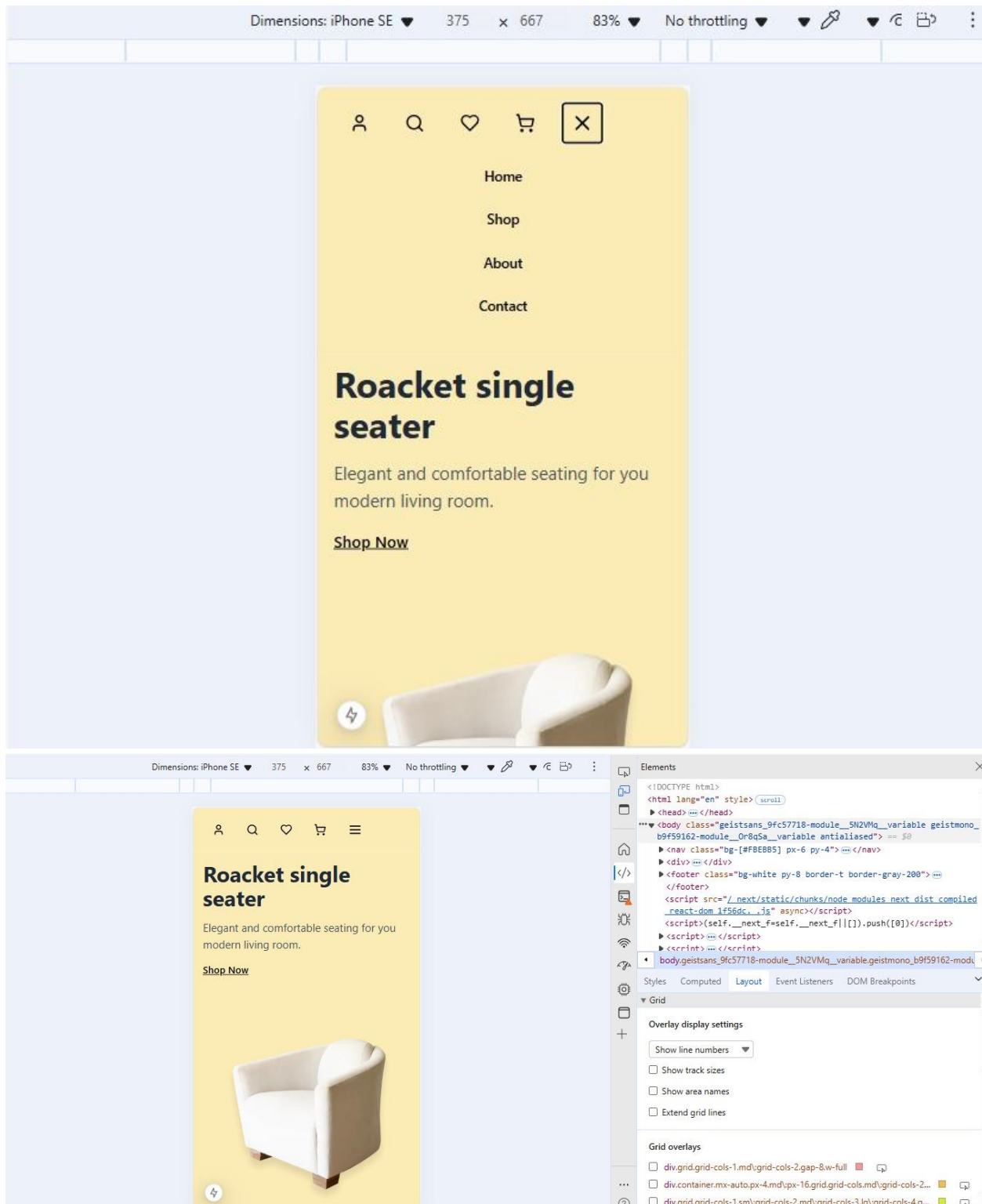
Cart operations

Add, update, and remove items from the cart working perfectly



Responsive

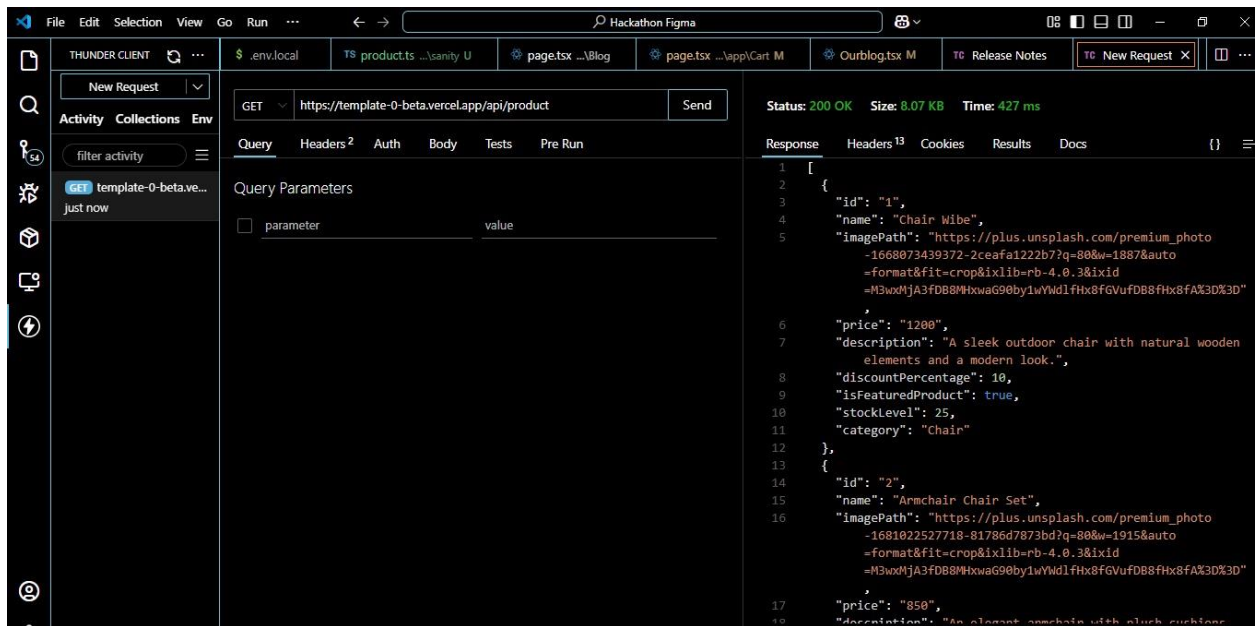
Responsive design testing has been completed to ensure the marketplace is fully optimized and functions seamlessly across all devices and screen sizes



Testing Tools

Postman

API responses have been tested using Postman to ensure they meet the expected behavior and performance standards



Performance Optimization

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	Show fallback UI with error message	Error message shown	Passed	Medium	-	Handled gracefully
4	TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	High	-	No issue found
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



Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0-49 ■ 50-89 ● 90-100



METRICS

Expand view

▲ First Contentful Paint
5.3 s

▲ Largest Contentful Paint
6.6 s



▲ Speed Index
9.9 s

View Treemap



Show audits relevant to: [All](#) [FCP](#) [LCP](#) [TBT](#) [CLS](#)

DIAGNOSTICS

- ▲ Minimize main-thread work — 11.1 s
- ▲ Largest Contentful Paint element — 6,550 ms
- ▲ Reduce JavaScript execution time — 1.9 s
- ▲ Largest Contentful Paint image was lazily loaded

PASSED AUDITS (21)

Show



Accessibility

These checks highlight opportunities to [improve the accessibility of your web app](#). Automatic detection can only detect a subset of issues and does not guarantee the accessibility of your web app, so [manual testing](#) is also encouraged.

NAMES AND LABELS

- ▲ Buttons do not have an accessible name

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive

100

Best Practices

TRUST AND SAFETY

○ Ensure CSP is effective against XSS attacks

○ Use a strong HSTS policy

○ Ensure proper origin isolation with COOP

GENERAL

▲ Missing source maps for large first-party JavaScript

http://localhost:3000/

339210092

100

Best Practices

TRUST AND SAFETY

○ Ensure CSP is effective against XSS attacks

○ Use a strong HSTS policy

○ Ensure proper origin isolation with COOP

Developer Resources

Dimensions: iPhone SE 375 x 667 83% No throttling

RoCKET single seater

Elegant and comfortable seating for you modern living room.

Shop Now

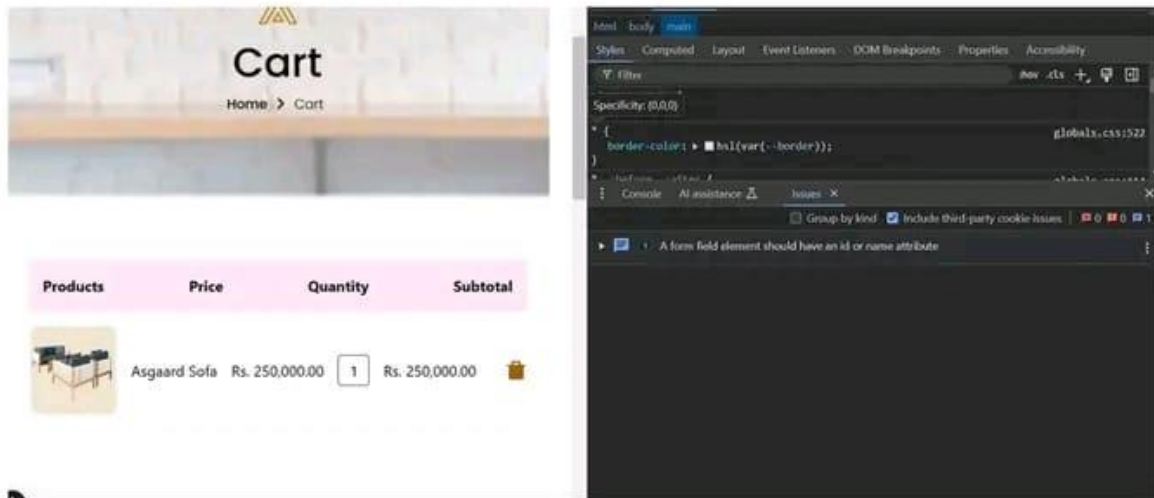
Developer resources

Filter by URL and error

☒ Load through website

Status	URL	Initiator	Total Bytes	Error
success	h./src_app_c7d265_.css.map	http/...	42 226	
success	/node_modules_%40swc_hel	http/...	5 502	
success	http://oc.../_a91c21_.js.map	http/...	149 998	
success	http://.../src_395c33_.js.map	http/...	10 431	
success	http://oc.../_824443_.js.map	http/...	134 762	
success	http://oc.../_d95469_.js.map	http/...	105 793	
success	/node_modules_next_dist_co	http/...	1 375 389	
success	/node_modules_next_dist_co	http/...	390 500	
success	/node_modules_next_dist_cli	http/...	1 054 548	
success	/node_modules_next_dist_4b	http/...	130 286	
success	/node_modules_699764_.js.r	http/...	1 162 462	
success	/%5BturboPACK%5D_browser	http/...	31 112	

ISSUES:



SECURITY:

