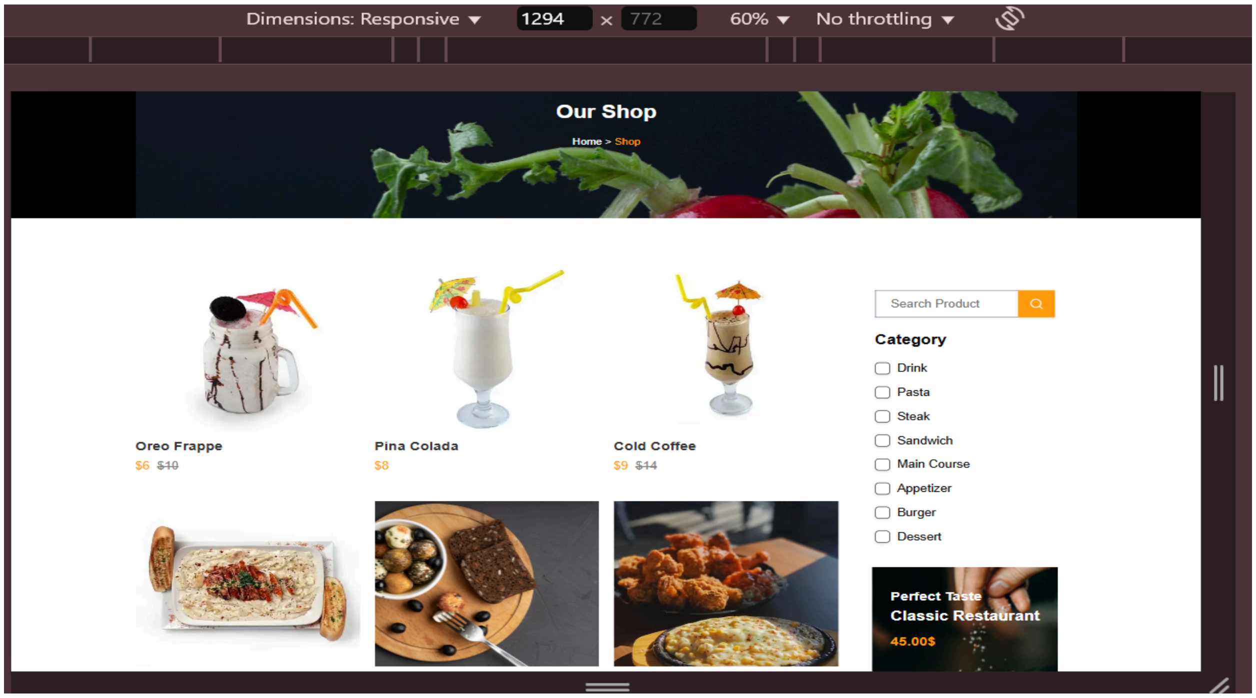
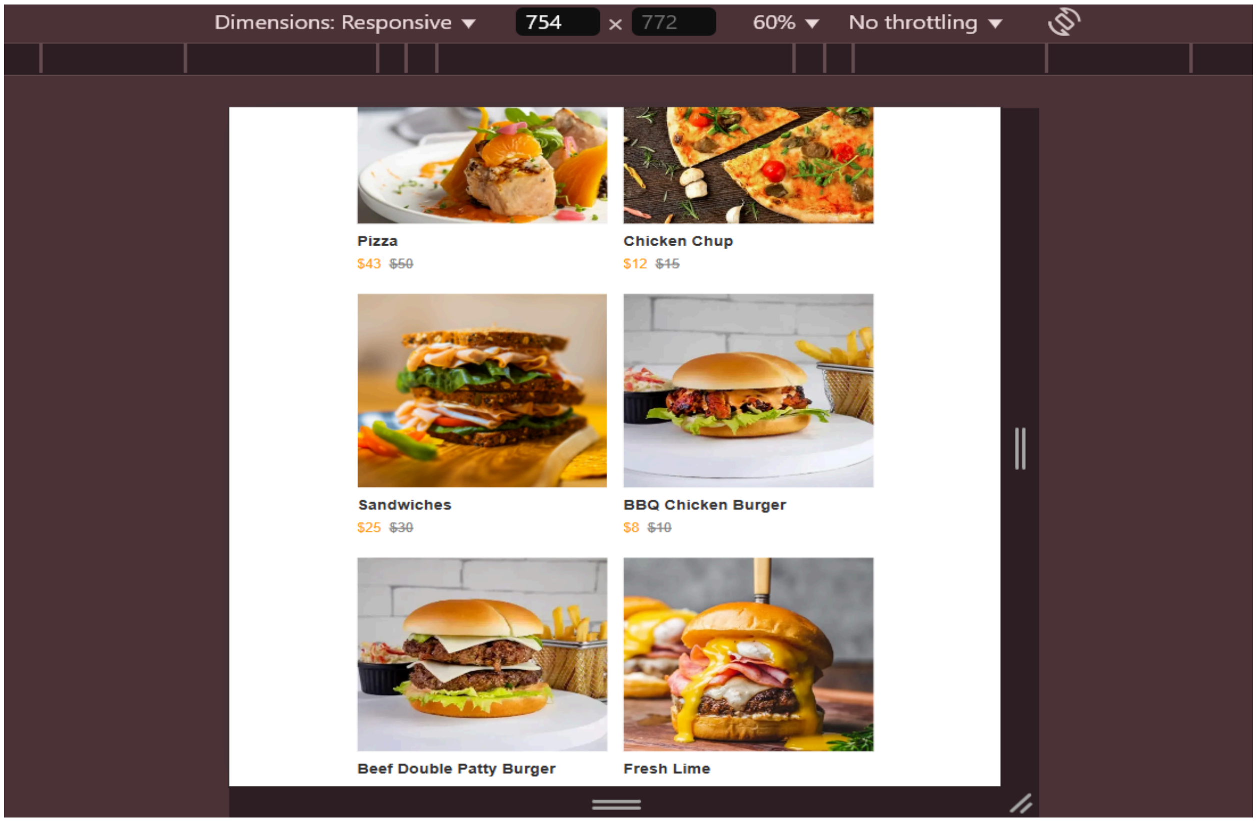
Table of Contents

Functional Testing

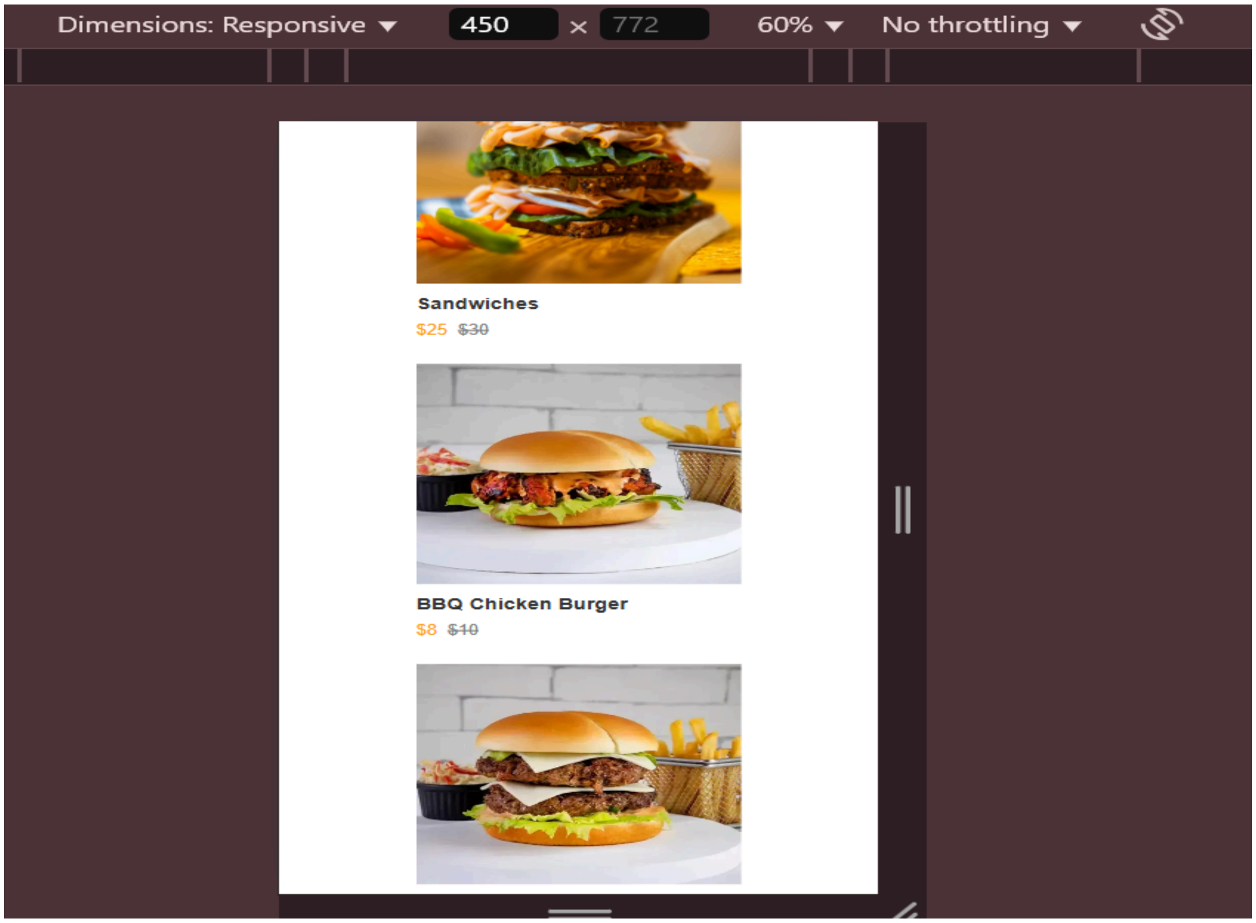
Product Listing Page with Functionality & Responsiveness



Dimensions: Responsive ▾ 1294 × 772 60% ▾ No throttling ▾



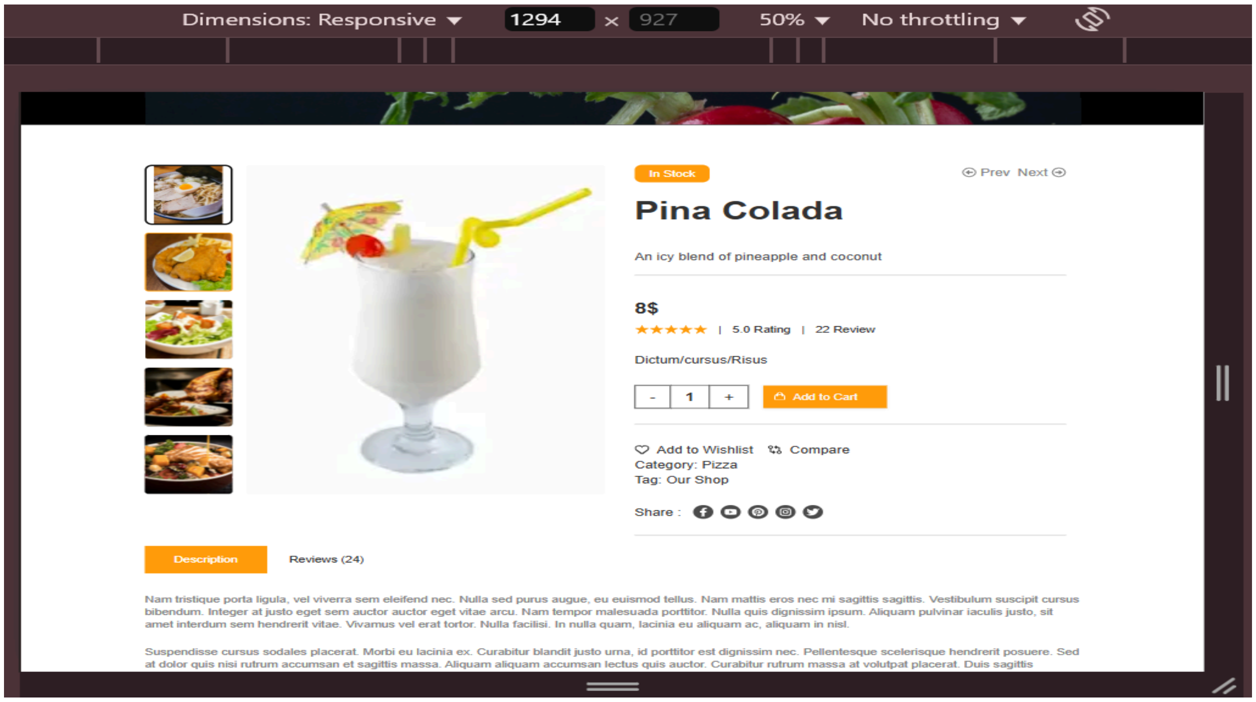
Dimensions: Responsive ▾ 754 × 772 60% ▾ No throttling ▾



Sandwiches
$25 ~~$30~~

BBQ Chicken Burger
$8 ~~$10~~

Product Detail Page with Functionality and Responsiveness



Pina Colada

An icy blend of pineapple and coconut

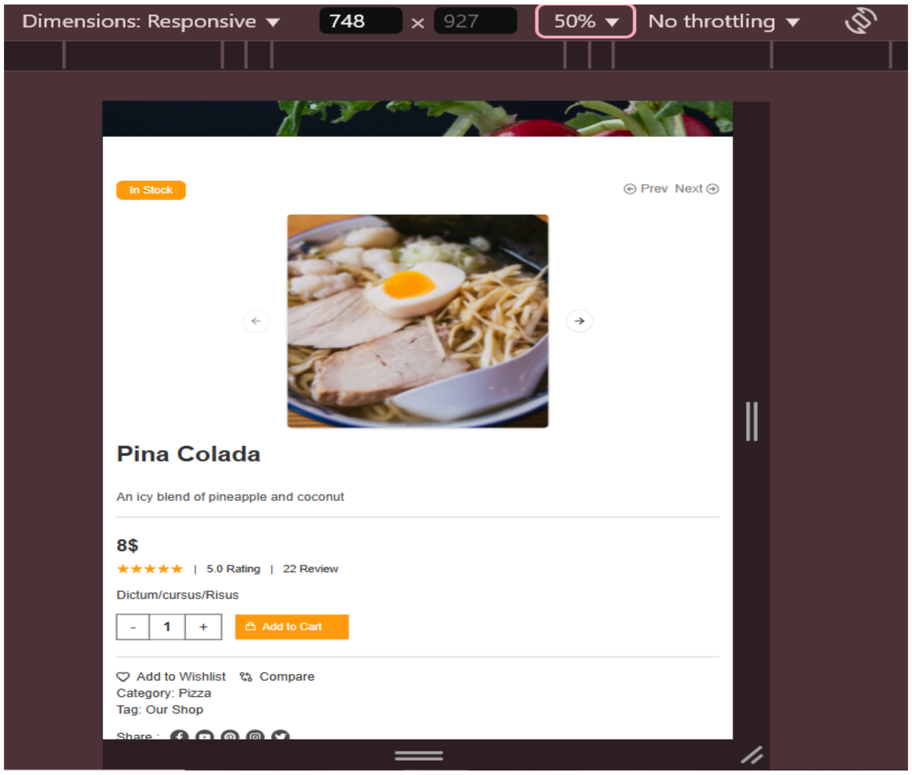
8$
★★★★★ | 5.0 Rating | 22 Review

Dictum/cursus/Risus

Category: Pizza
Tag: Our Shop

Nam tristique porta ligula, vel viverra sem eleifend nec. Nulla sed purus augue, eu euismod tellus. Nam mattis eros nec mi sagittis sagittis. Vestibulum suscipit cursus bibendum. Integer at justo eget sem auctor auctor eget vitae arcu. Nam tempor malesuada porttitor. Nulla quis dignissim ipsum. Aliquam pulvinar iaculis justo, sit amet interdum sem hendrerit vitae. Vivamus vel erat tortor. Nulla facilisi. In nulla quam, lacinia eu aliquam ac, aliquam in nisl.

Suspendisse cursus sodales placerat. Morbi eu lacinia ex. Curabitur blandit justo urna, id porttitor est dignissim nec. Pellentesque scelerisque hendrerit posuere. Sed at dolor quis nisi rutrum accumsan et sagittis massa. Aliquam aliquam accumsan lectus quis auctor. Curabitur rutrum massa at volutpat placerat. Duis sagittis



Dimensions: Responsive 748 × 927 50% No throttling

In Stock

Prev Next

Pina Colada

An icy blend of pineapple and coconut

8$

★★★★★ | 5.0 Rating | 22 Review

Dictum/cursus/Risus

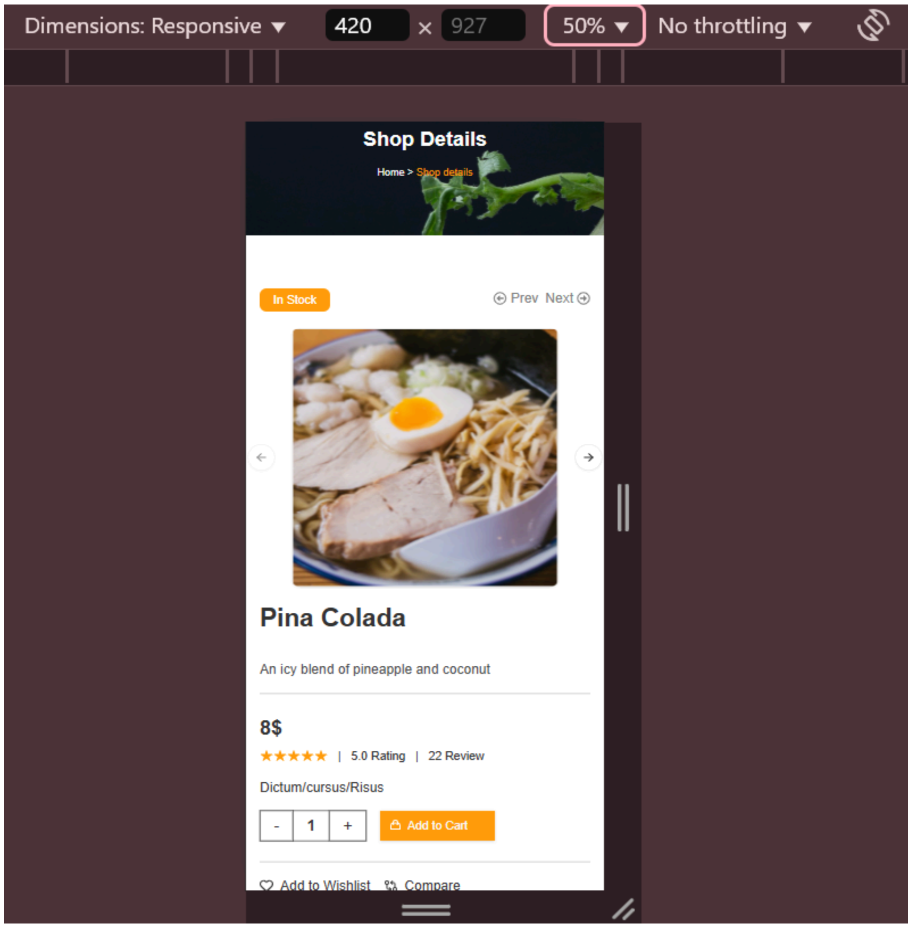
- 1 + Add to Cart

Add to Wishlist Compare

Category: Pizza

Tag: Our Shop

Share :



Dimensions: Responsive 420 × 927 50% No throttling

Shop Details

Home > Shop details

In Stock

Prev Next

Pina Colada

An icy blend of pineapple and coconut

8$

★★★★★ | 5.0 Rating | 22 Review

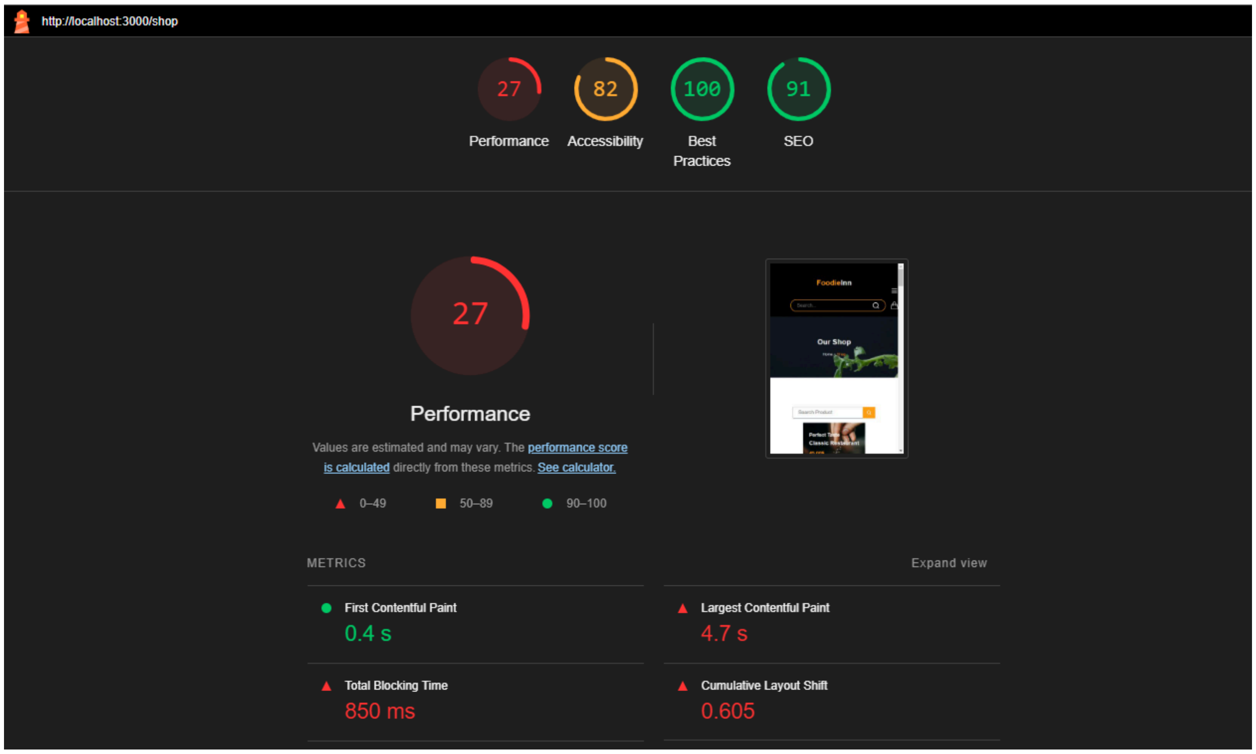
Dictum/cursus/Risus

- 1 + Add to Cart

Add to Wishlist Compare

Logs from Testing Tools

Lighthouse



http://localhost:3000/shop

27	82	100	91
Performance	Accessibility	Best Practices	SEO

27

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

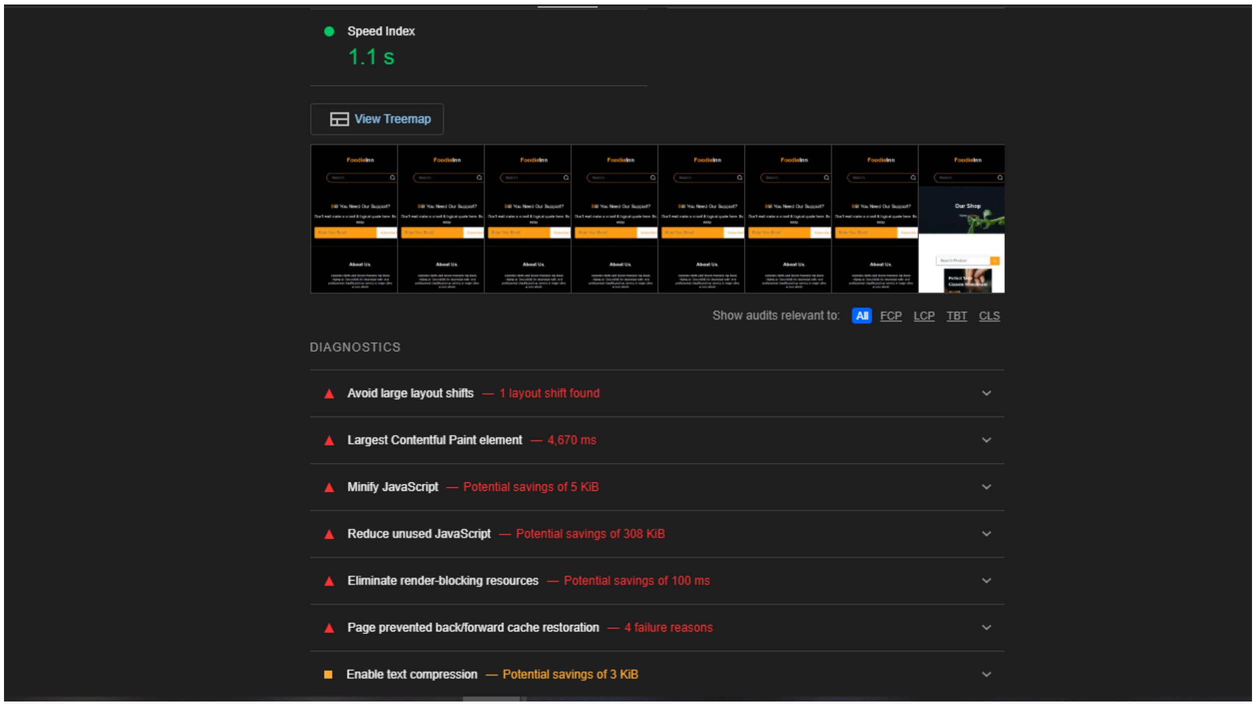
METRICS Expand view

● First Contentful Paint
0.4 s

▲ Largest Contentful Paint
4.7 s

▲ Total Blocking Time
850 ms

▲ Cumulative Layout Shift
0.605



● Speed Index
1.1 s

View Treemap

Show audits relevant to: All FCP LCP TBT CLS

DIAGNOSTICS

▲ Avoid large layout shifts — 1 layout shift found

▲ Largest Contentful Paint element — 4,670 ms

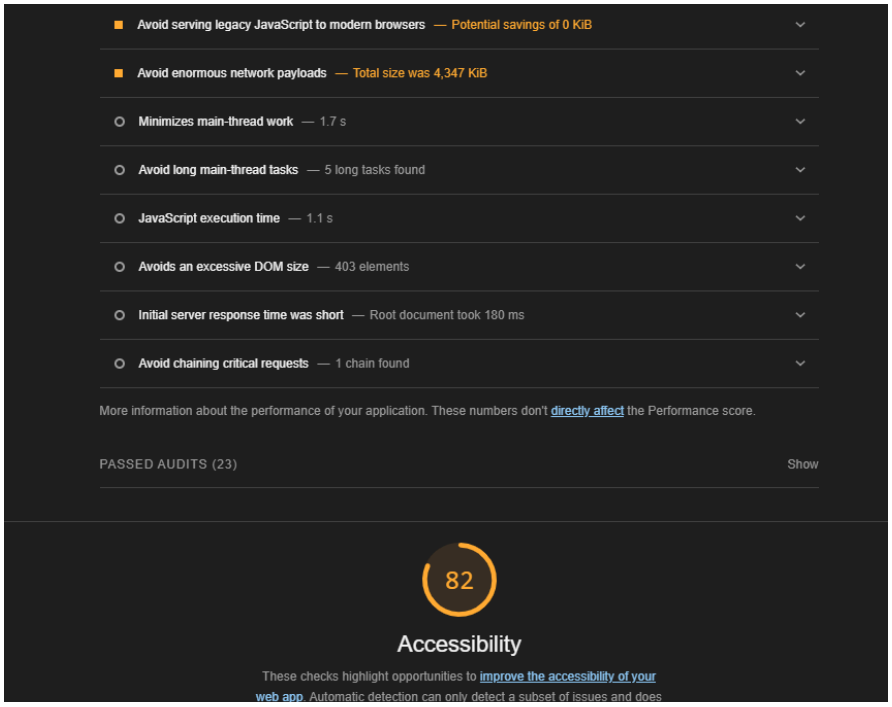
▲ Minify JavaScript — Potential savings of 5 KiB

▲ Reduce unused JavaScript — Potential savings of 308 KiB

▲ Eliminate render-blocking resources — Potential savings of 100 ms

▲ Page prevented back/forward cache restoration — 4 failure reasons

■ Enable text compression — Potential savings of 3 KiB



■ Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB

■ Avoid enormous network payloads — Total size was 4,347 KiB

○ Minimizes main-thread work — 1.7 s

○ Avoid long main-thread tasks — 5 long tasks found

○ JavaScript execution time — 1.1 s

○ Avoids an excessive DOM size — 403 elements

○ Initial server response time was short — Root document took 180 ms

○ Avoid chaining critical requests — 1 chain found

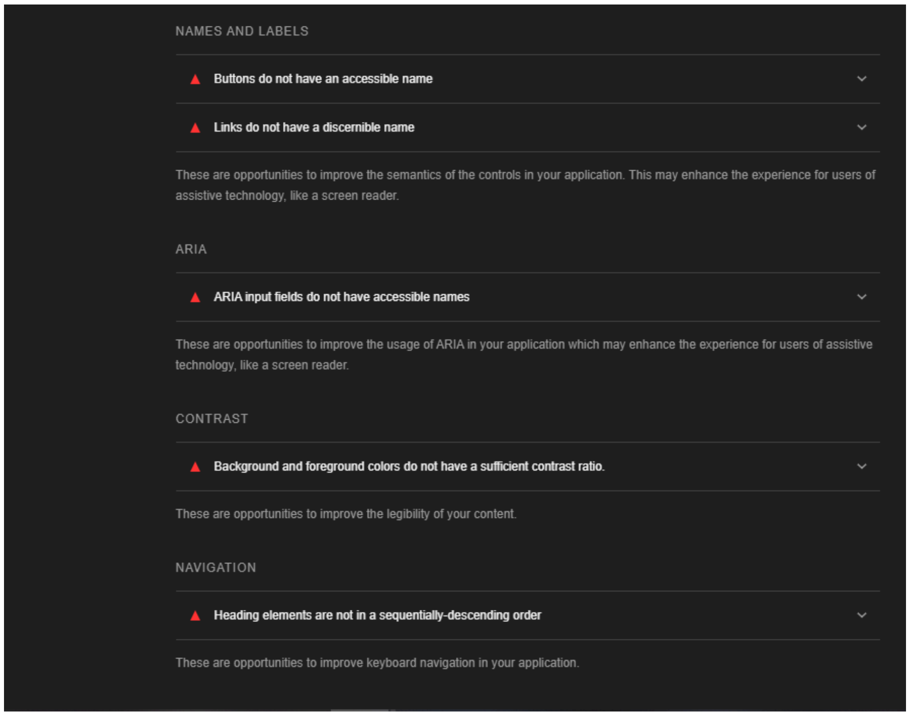
More information about the performance of your application. These numbers don't directly affect the Performance score.

PASSED AUDITS (23) Show

82

Accessibility

These checks highlight opportunities to improve the accessibility of your web app. Automatic detection can only detect a subset of issues and does



NAMES AND LABELS

▲ Buttons do not have an accessible name

▲ Links do not have a discernible name

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, like a screen reader.

ARIA

▲ ARIA input fields do not have accessible names

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, like a screen reader.

CONTRAST

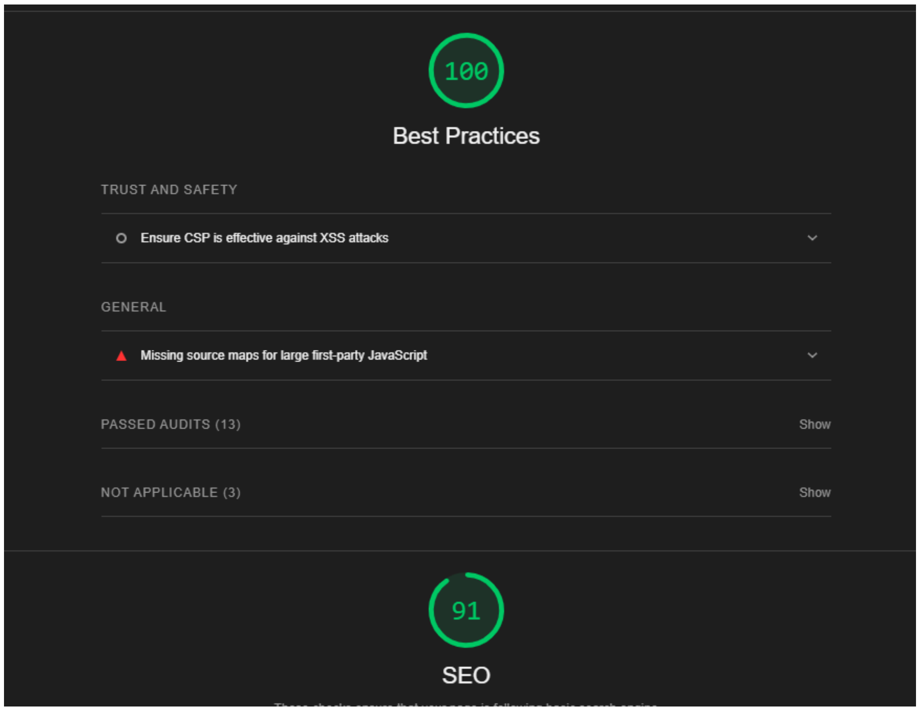
▲ Background and foreground colors do not have a sufficient contrast ratio.

These are opportunities to improve the legibility of your content.

NAVIGATION

▲ Heading elements are not in a sequentially-descending order

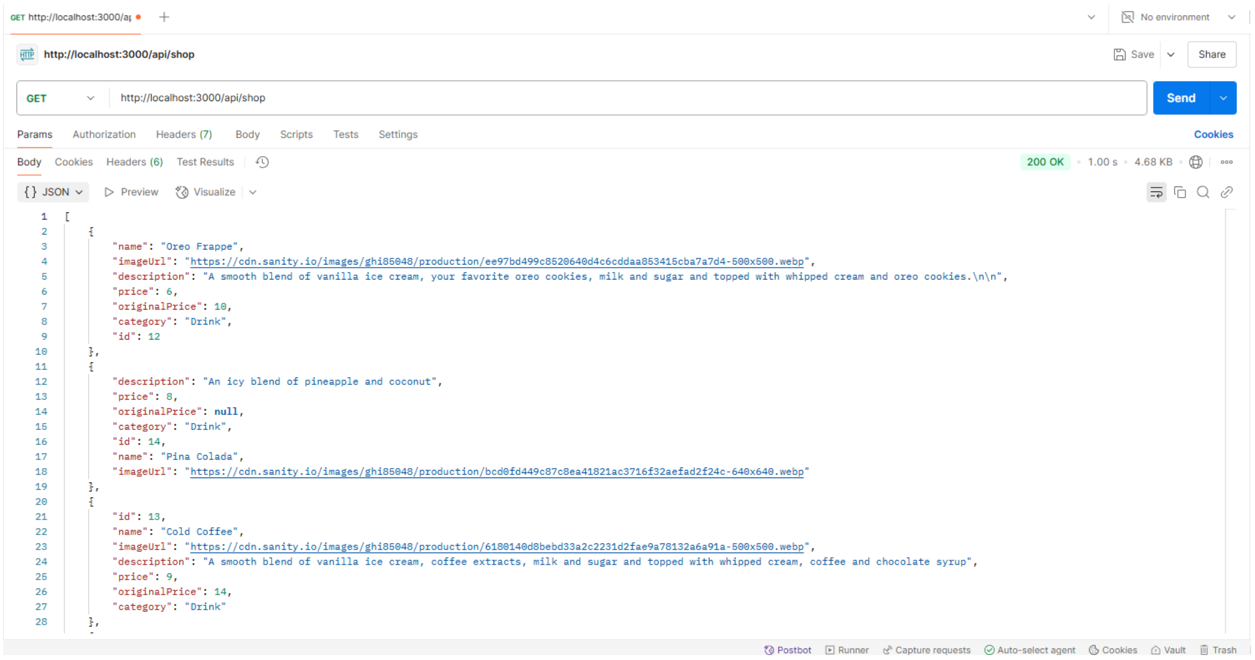
These are opportunities to improve keyboard navigation in your application.



100
Best Practices
TRUST AND SAFETY
Ensure CSP is effective against XSS attacks
GENERAL
Missing source maps for large first-party JavaScript
PASSED AUDITS (13)
Show
NOT APPLICABLE (3)
Show
91
SEO

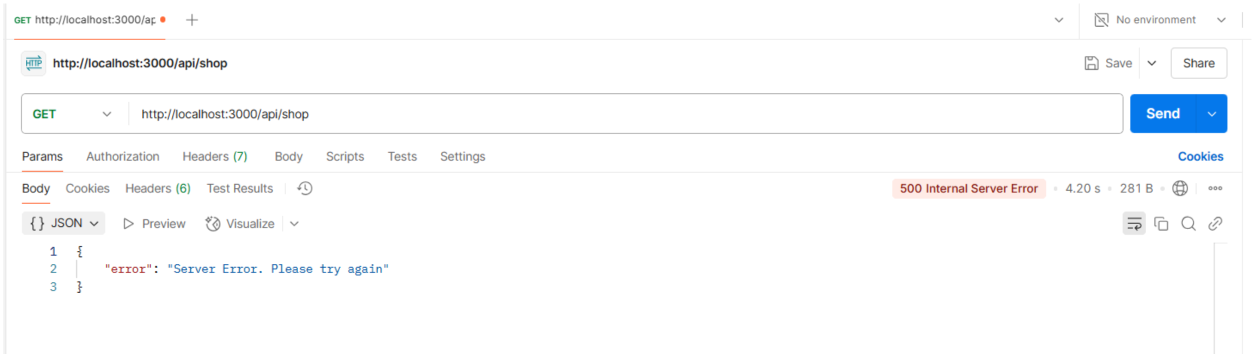
Postman

Successful fetching of all products



```
[
    {
        "name": "Oreo Frappe",
        "imageUrl": "https://cdn.sanity.io/images/ghi85048/production/ee97bd499c8520640d4c6cddaa853415cba7a7d4-500x500.webp",
        "description": "A smooth blend of vanilla ice cream, your favorite oreo cookies, milk and sugar and topped with whipped cream and oreo cookies.\n\n",
        "price": 6,
        "originalPrice": 10,
        "category": "Drink",
        "id": 12
    },
    {
        "description": "An icy blend of pineapple and coconut",
        "price": 8,
        "originalPrice": null,
        "category": "Drink",
        "id": 14,
        "name": "Pina Colada",
        "imageUrl": "https://cdn.sanity.io/images/ghi85048/production/bcd0fd449c87c8ea41821ac3716f32aefad2f24c-640x640.webp"
    },
    {
        "id": 13,
        "name": "Cold Coffee",
        "imageUrl": "https://cdn.sanity.io/images/ghi85048/production/6180140d8bebd33a2c2231d2fae9a78132a6a91a-500x500.webp",
        "description": "A smooth blend of vanilla ice cream, coffee extracts, milk and sugar and topped with whipped cream, coffee and chocolate syrup",
        "price": 9,
        "originalPrice": 14,
        "category": "Drink"
    },
```

Error Handling for status 500


GET http://localhost:3000/ap
No environment
Save
Share
Send
Params Authorization Headers (7) Body Scripts Tests Settings
Cookies
Body Cookies Headers (6) Test Results
JSON Preview Visualize

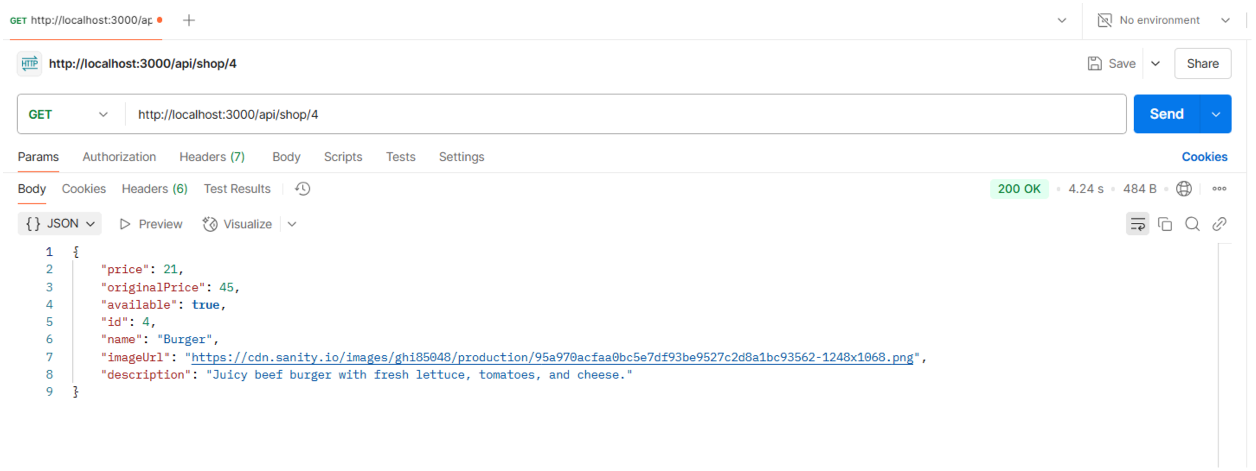
http://localhost:3000/api/shop

GET http://localhost:3000/api/shop

500 Internal Server Error · 4.20 s · 281 B

```
{
  "error": "Server Error. Please try again"
}
```

Successful fetching of individual product


GET http://localhost:3000/ap
No environment
Save
Share
Send
Params Authorization Headers (7) Body Scripts Tests Settings
Cookies
Body Cookies Headers (6) Test Results
JSON Preview Visualize

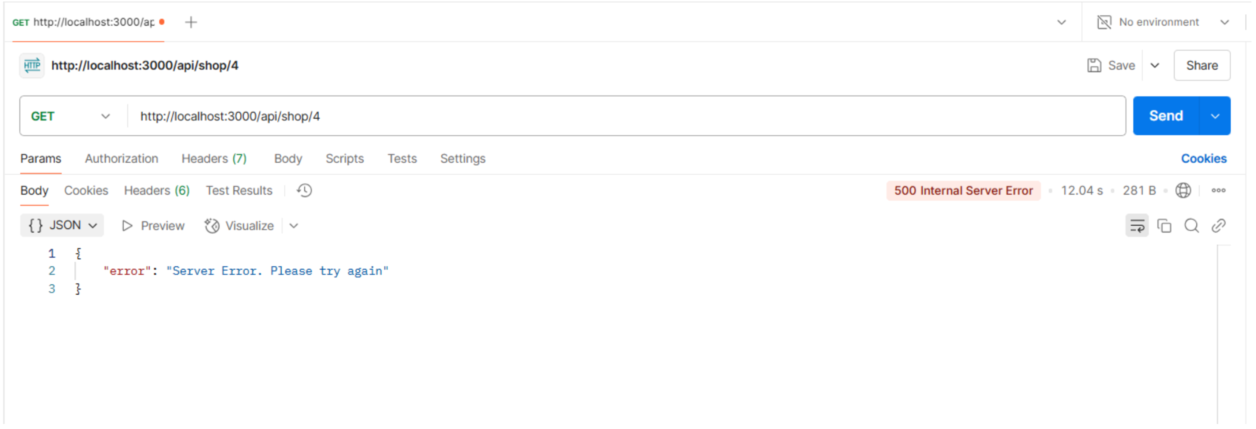
http://localhost:3000/api/shop/4

GET http://localhost:3000/api/shop/4

200 OK · 4.24 s · 484 B

```
{
  "price": 21,
  "originalPrice": 45,
  "available": true,
  "id": 4,
  "name": "Burger",
  "imageUrl": "https://cdn.sanity.io/images/ghi85048/production/95a970acfaa0bc5e7df93be9527c2d8a1bc93562-1248x1068.png",
  "description": "Juicy beef burger with fresh lettuce, tomatoes, and cheese."
}
```

Error Handling for status 500


GET http://localhost:3000/ap
No environment
Save
Share
Send
Params Authorization Headers (7) Body Scripts Tests Settings
Cookies
Body Cookies Headers (6) Test Results
JSON Preview Visualize

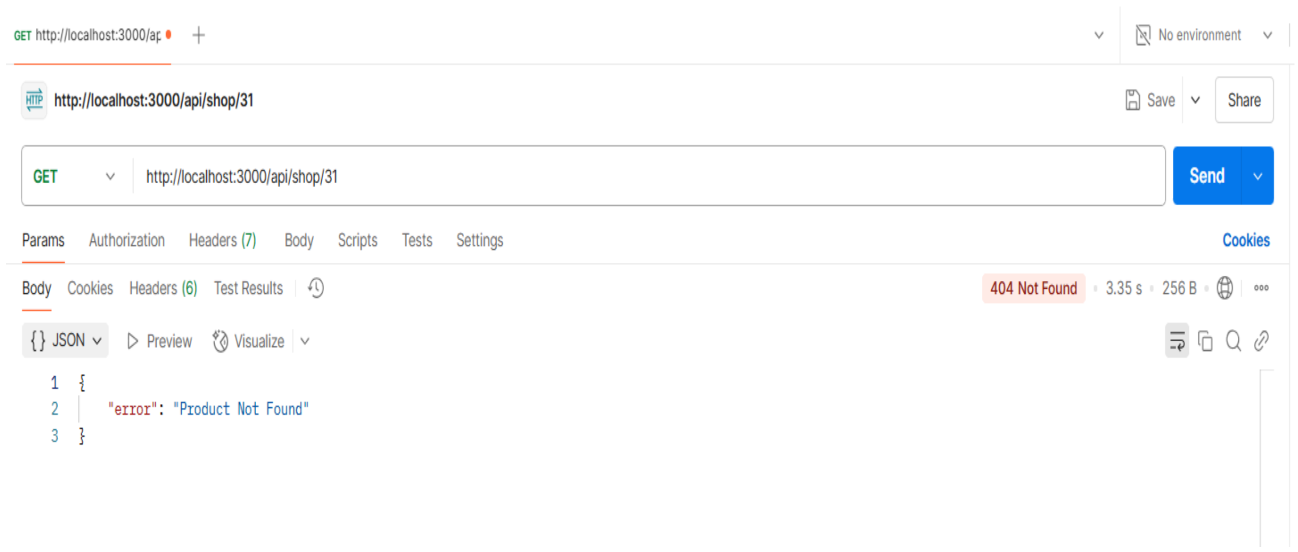
http://localhost:3000/api/shop/4

GET http://localhost:3000/api/shop/4

500 Internal Server Error · 12.04 s · 281 B

```
{
  "error": "Server Error. Please try again"
}
```

Error Handling for status 404 - Not Found



http://localhost:3000/api/shop/31

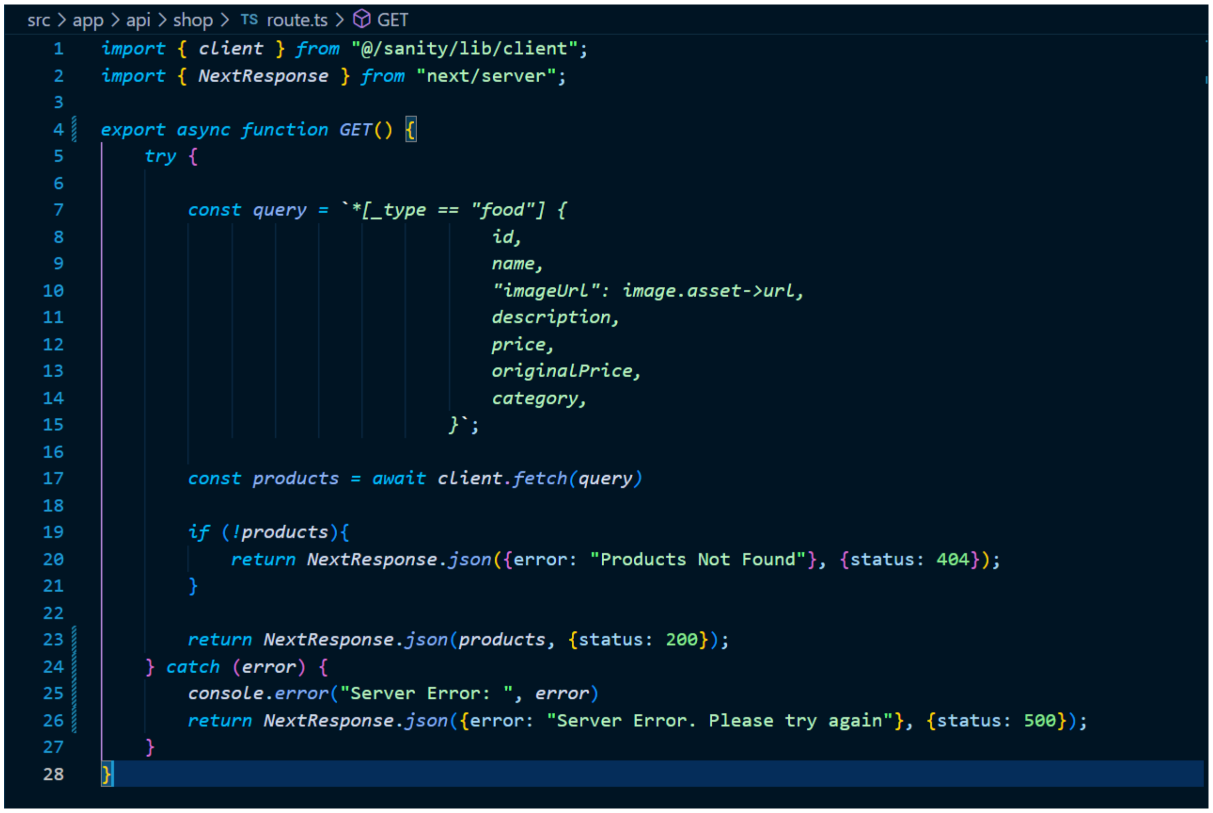
GET http://localhost:3000/api/shop/31

404 Not Found

```
{
    "error": "Product Not Found"
}
```

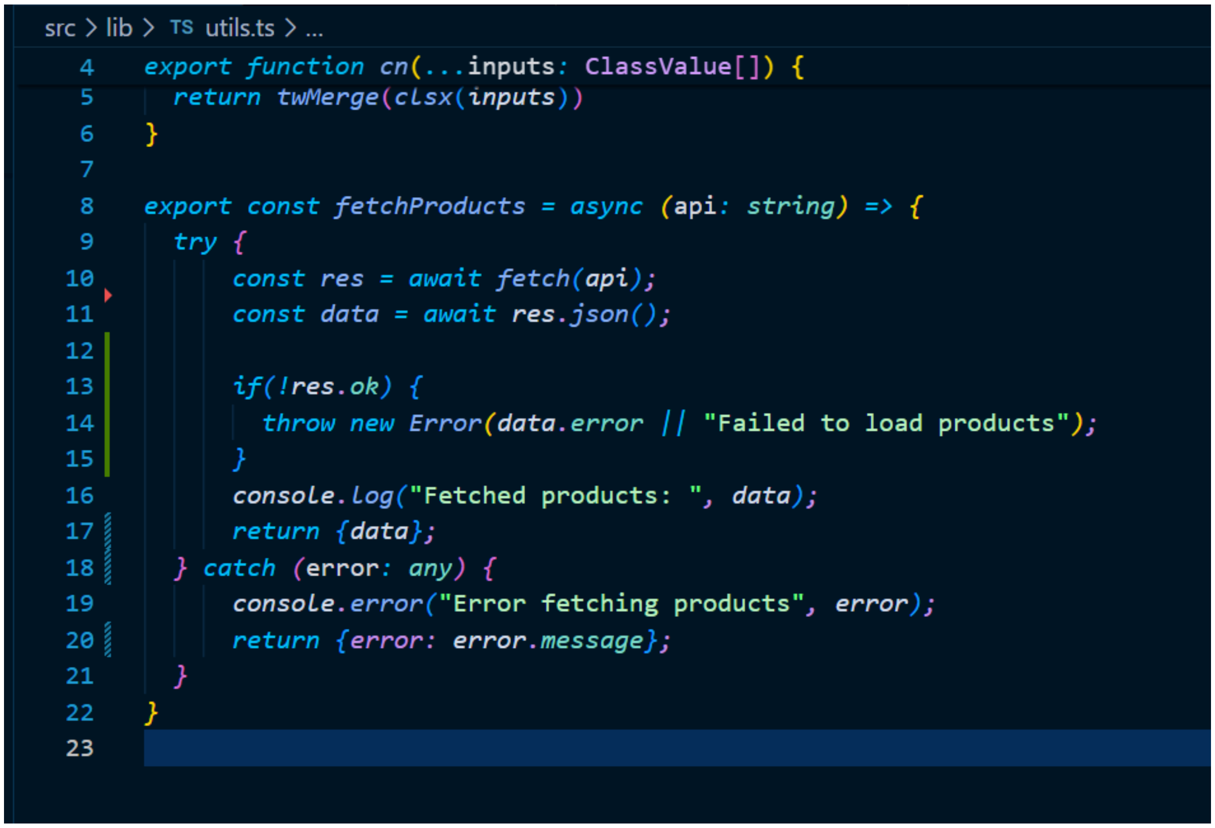
Error Handling

In my project, I implemented robust error handling to ensure smooth functionality and a better user experience. I used **try-catch** blocks in API route handlers to catch and handle errors gracefully. For frontend error management, I incorporated an **error fallback UI**, allowing users to see meaningful error messages instead of application crashes. Additionally, I structured my error handling in a **modular and scalable** way, making it easier to maintain and extend. Logging mechanisms were also added to track unexpected errors and improve debugging.



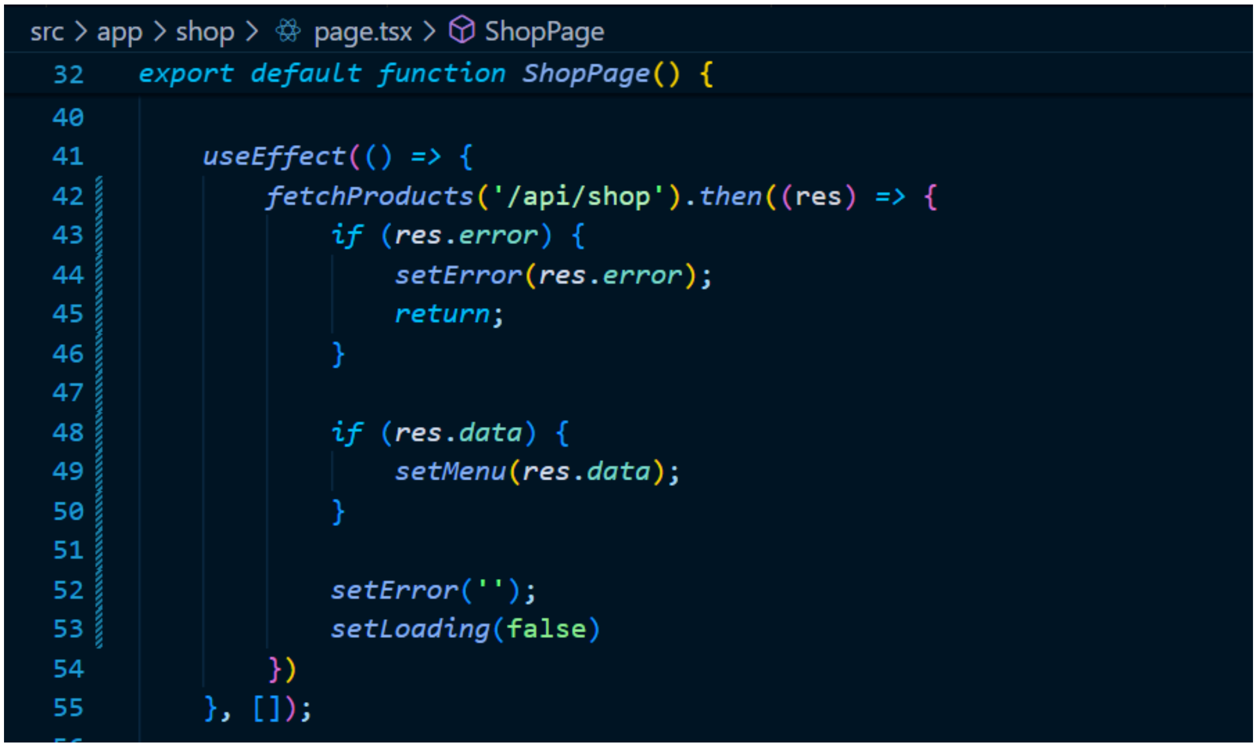
src > app > api > shop > TS route.ts > GET

```
import { client } from "@/sanity/lib/client";
import { NextResponse } from "next/server";

export async function GET() {
    try {

        const query = `*[_type == "food"] {
                            id,
                            name,
                            "imageUrl": image.asset->url,
                            description,
                            price,
                            originalPrice,
                            category,
                        }`;

        const products = await client.fetch(query)

        if (!products){
            return NextResponse.json({error: "Products Not Found"}, {status: 404});
        }

        return NextResponse.json(products, {status: 200});
    } catch (error) {
        console.error("Server Error: ", error)
        return NextResponse.json({error: "Server Error. Please try again"}, {status: 500});
    }
}
```



src > lib > TS utils.ts > ...

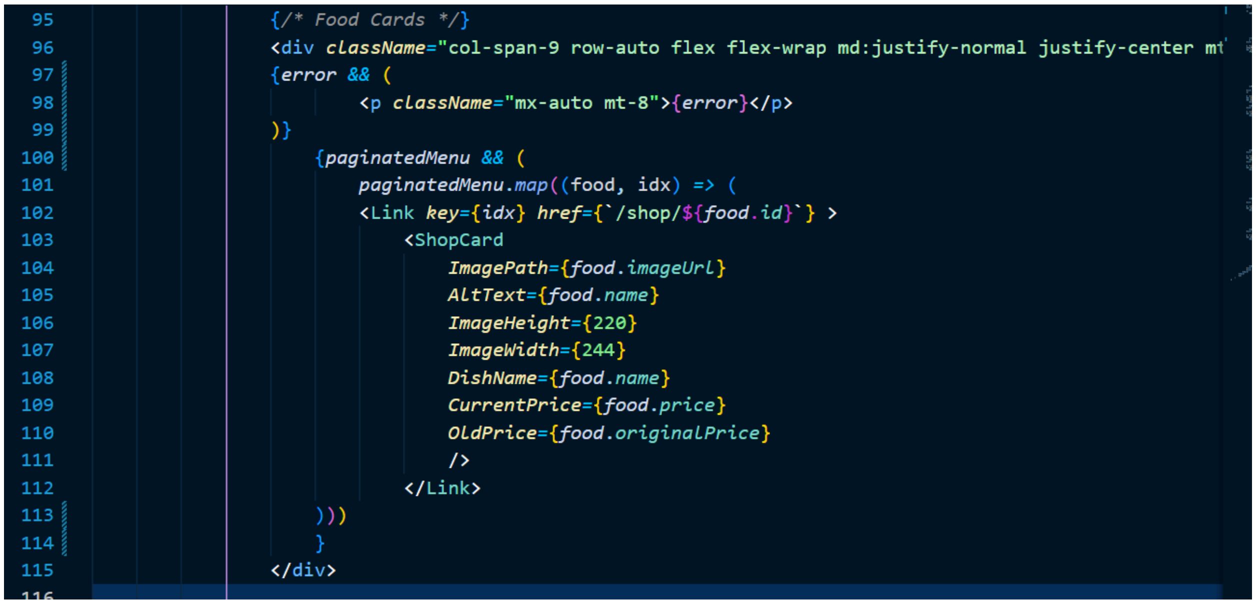
```
export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}

export const fetchProducts = async (api: string) => {
  try {
    const res = await fetch(api);
    const data = await res.json();

    if(!res.ok) {
      throw new Error(data.error || "Failed to load products");
    }
    console.log("Fetched products: ", data);
    return {data};
  } catch (error: any) {
    console.error("Error fetching products", error);
    return {error: error.message};
  }
}
```



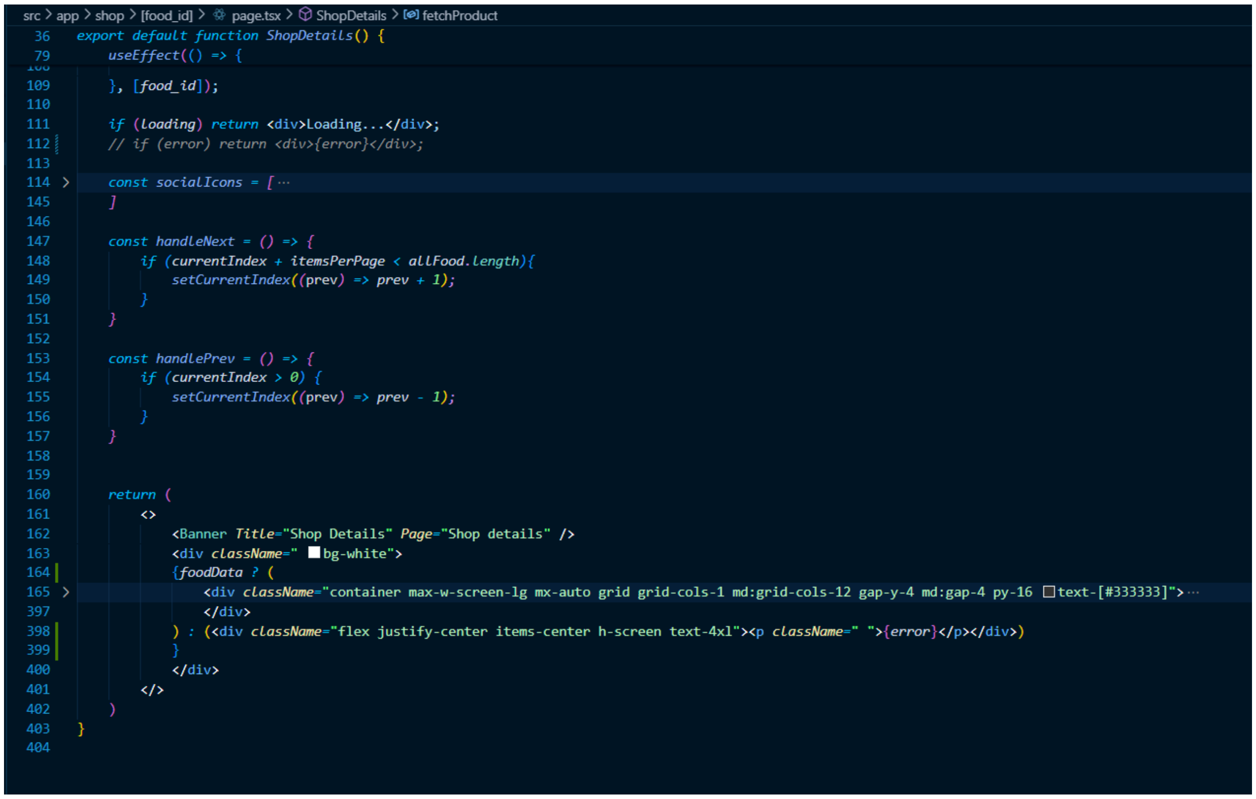
src > app > shop > page.tsx > ShopPage

```
export default function ShopPage() {

    useEffect(() => {
        fetchProducts('/api/shop').then((res) => {
            if (res.error) {
                setError(res.error);
                return;
            }

            if (res.data) {
                setMenu(res.data);
            }

            setError('');
            setLoading(false)
        })
    }, []);
```

Fallback UI Examples



```
{/* Food Cards */}
<div className="col-span-9 row-auto flex flex-wrap md:justify-normal justify-center mt
{error && (
        <p className="mx-auto mt-8">{error}</p>
)}
    {paginatedMenu && (
        paginatedMenu.map((food, idx) => (
        <Link key={idx} href={`/shop/${food.id}`} >
            <ShopCard
                ImagePath={food.imageUrl}
                AltText={food.name}
                ImageHeight={220}
                ImageWidth={244}
                DishName={food.name}
                CurrentPrice={food.price}
                OldPrice={food.originalPrice}
                />
            </Link>
    )))
    }
</div>
```


src > app > shop > [food_id] > page.tsx > ShopDetails > fetchProduct

```
export default function ShopDetails() {
    useEffect(() => {
    }, [food_id]);

    if (loading) return <div>Loading...</div>;
    // if (error) return <div>{error}</div>;

    const socialIcons = [⋯
    ]

    const handleNext = () => {
        if (currentIndex + itemsPerPage < allFood.length){
            setCurrentIndex((prev) => prev + 1);
        }
    }

    const handlePrev = () => {
        if (currentIndex > 0) {
            setCurrentIndex((prev) => prev - 1);
        }
    }


    return (
        <>
            <Banner Title="Shop Details" Page="Shop details" />
            <div className=" bg-white">
            {foodData ? (
                <div className="container max-w-screen-lg mx-auto grid grid-cols-1 md:grid-cols-12 gap-y-4 md:gap-4 py-16 text-[#333333]">⋯
                </div>
            ) : (<div className="flex justify-center items-center h-screen text-4xl"><p className=" ">{error}</p></div>)
            }
            </div>
        </>
    )
}
```

Cross-Browser & Device Testing

Description:

I tested the marketplace across various browsers and devices to ensure consistent performance and responsiveness.

Key Points:

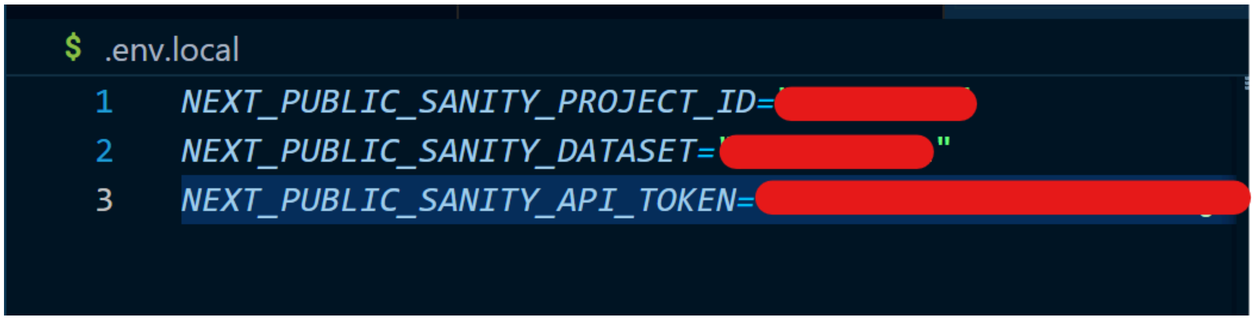
- Tested on popular browsers: Chrome, FireFox, Safari, and Microsoft Edge.
- Verified responsiveness on desktop, tablet, and mobile devices.
- Ensured no layout issues or broken features across different screen sizes.

Security Testing

Description:
I prioritized securing the marketplace by implementing measures to ensure safe communication and protect sensitive data.

Secure API Communication:

- Store sensitive data like API keys in environment variables to prevent exposure.



```
$ .env.local
NEXT_PUBLIC_SANITY_PROJECT_ID=[redacted]
NEXT_PUBLIC_SANITY_DATASET=[redacted]"
NEXT_PUBLIC_SANITY_API_TOKEN=[redacted]
```

User Acceptance Testing (UAT)

Description:
I tested the marketplace to ensure it meets real-world usage expectations.

Key Points:

- Simulated tasks like browsing, navigating between different pages, search, and categorization.
- Collected feedback from peers to improve usability.

Expected Result:
A seamless and user-friendly experience.

ActualResult:
UAT completed successfully with no major issues.

Final Checklist

Task	Status
Functional Testing	☑
Performance Testing	☑
Error Handling	☑
Device Testing	☑
Security Testing	☑
Documentation	☑