

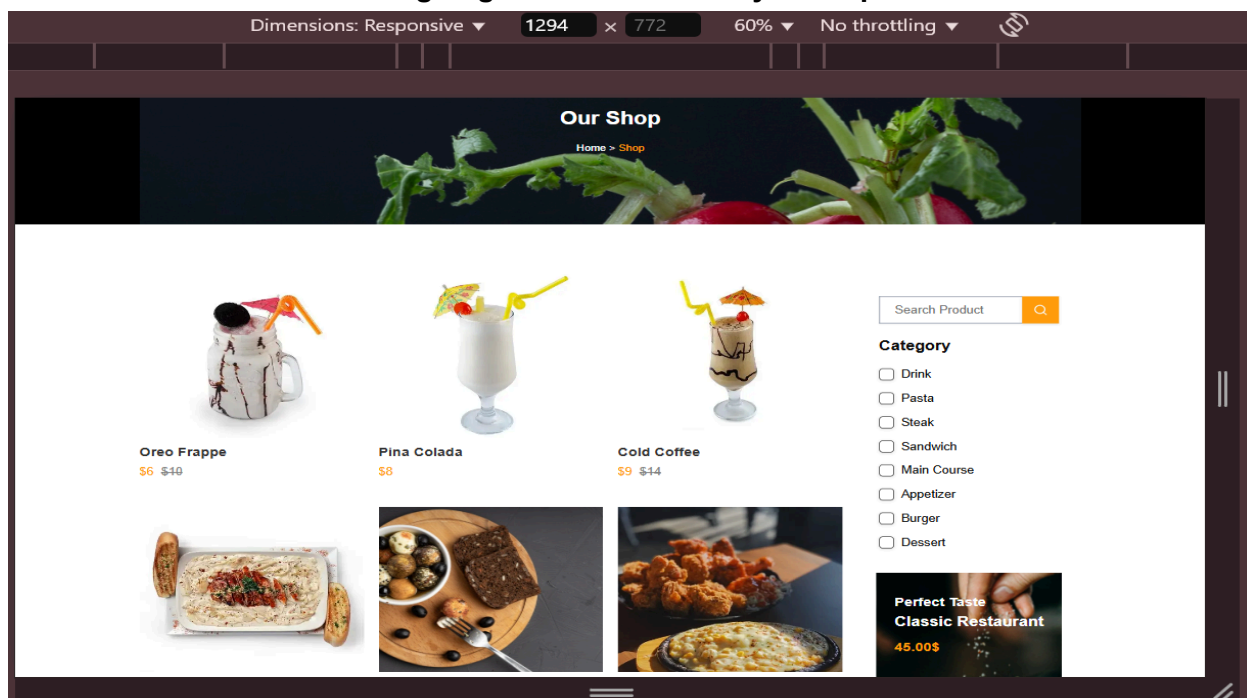
Table of Contents

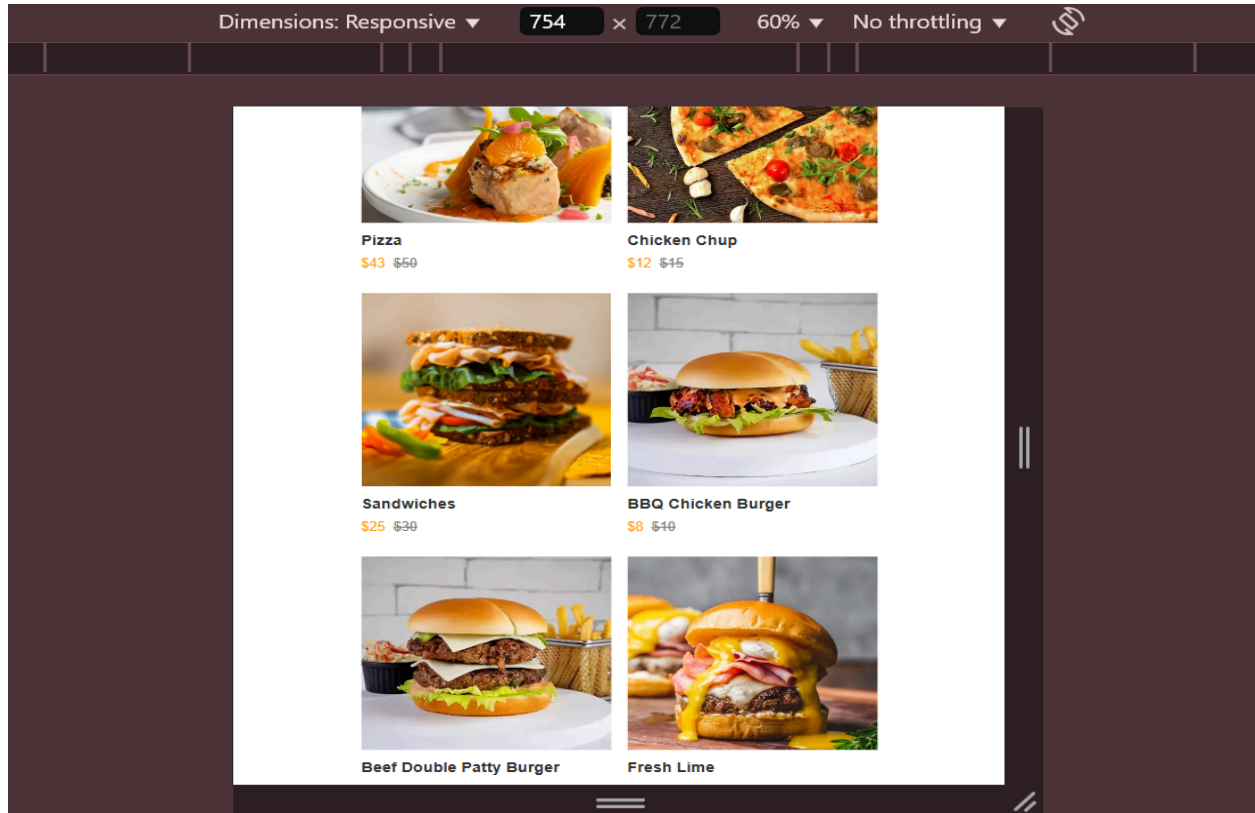
Functional Testing	1
• Product Listing Page with Functionality & Responsiveness	1
• Product Detail Page with Functionality & Responsiveness	2
• Cart Operations	4
• Logs from Testing Tools	5
◦ Lighthouse	4
◦ Postman	8
Error Handling	10
• Fallback UI Examples	12
Cross-Browser & Device Testing	13
Security Testing	13
User Acceptance Testing (UAT)	14
Final Checklist	14

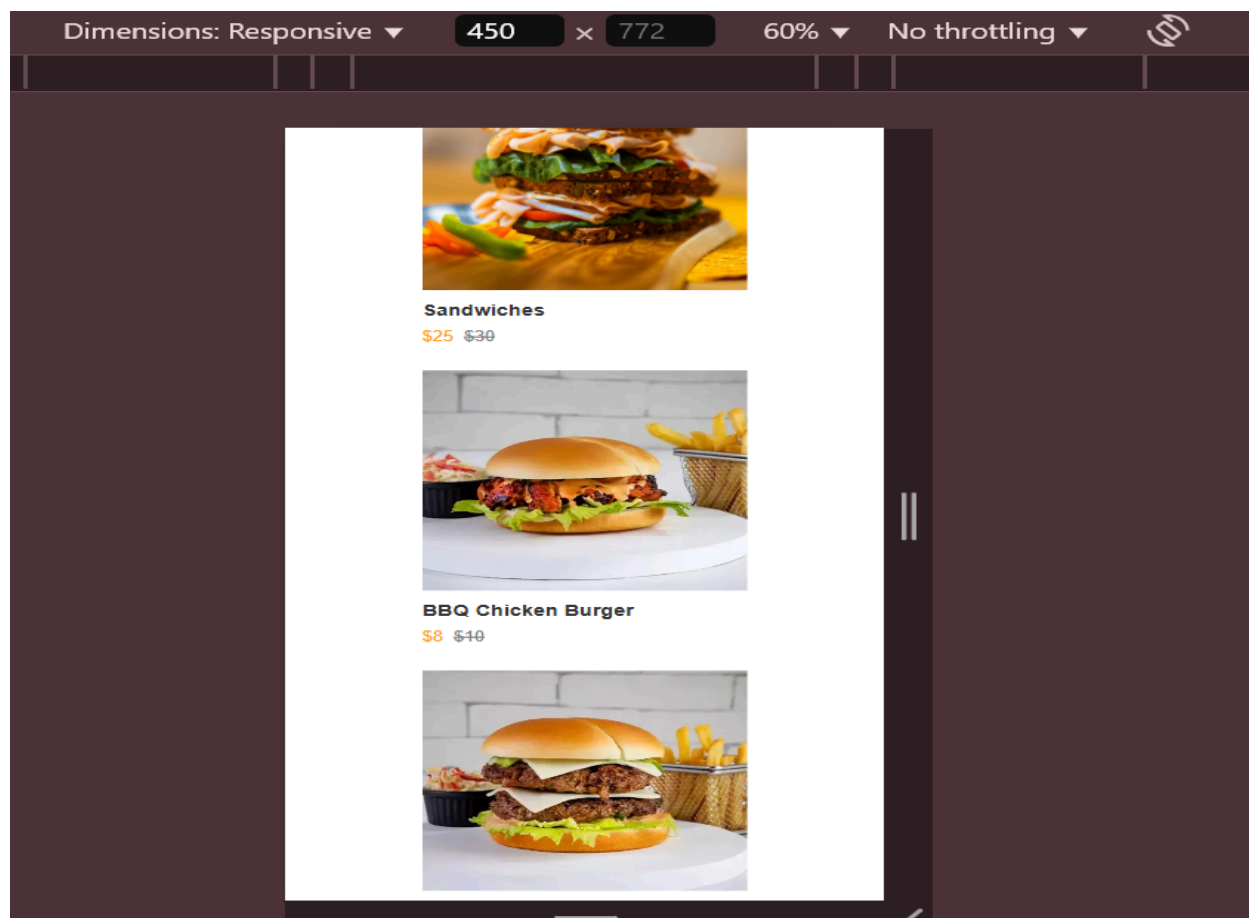
1.0	Without cart functionality screenshots
1.1	With cart functionality screenshots

Functional Testing

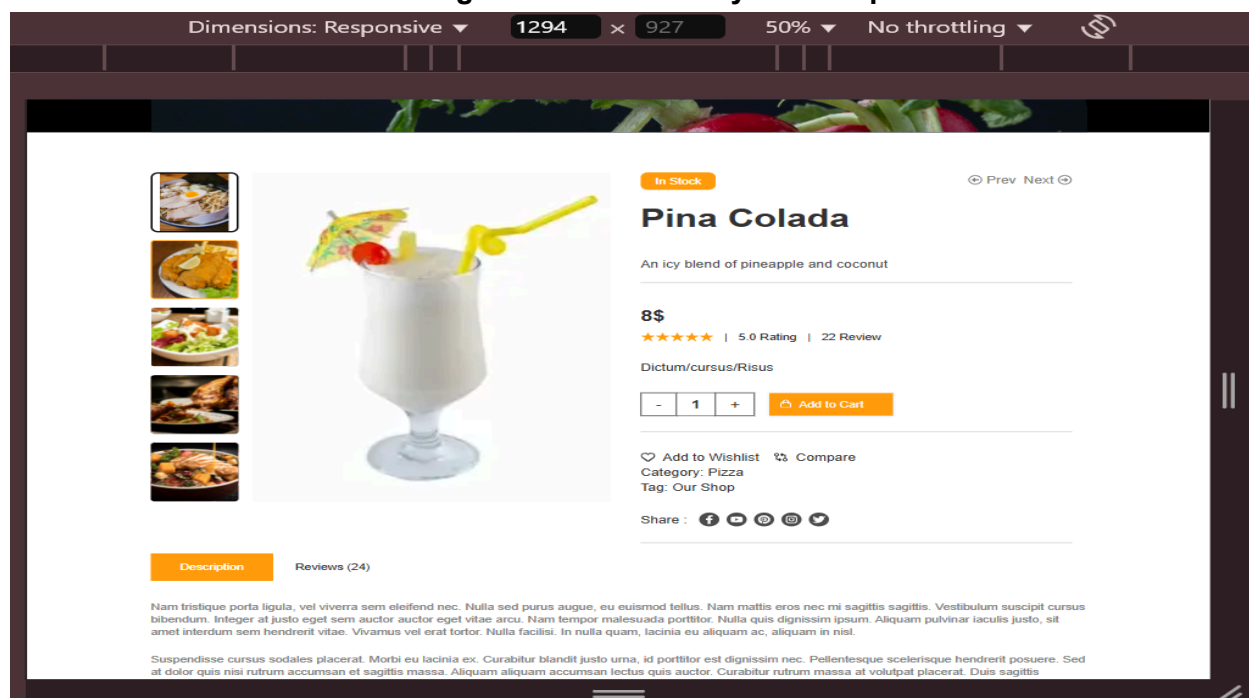
Product Listing Page with Functionality & Responsiveness

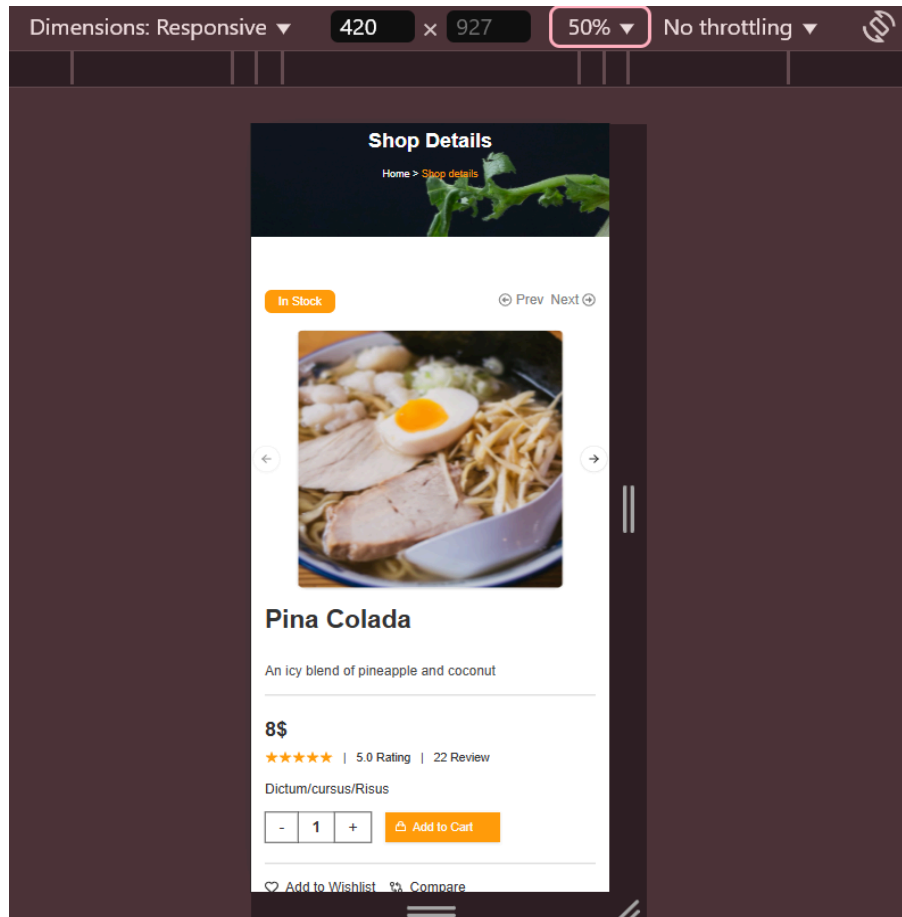
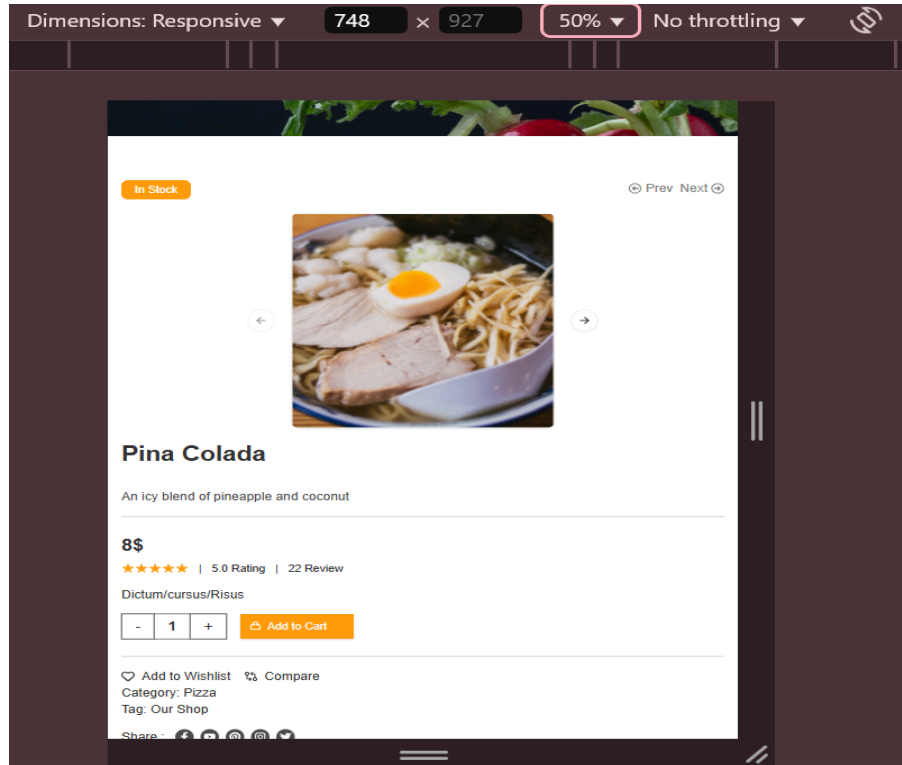






Product Detail Page with Functionality and Responsiveness





Cart Operations

Add

☐ Hide network
 ☐ Log XMLHttpRequests

☐ Preserve log
 ☒ Eager evaluation

☐ Selected context only
 ☒ Autocomplete from history

☒ Group similar messages in console
 ☒ Treat code evaluation as user action

☒ Show CORS errors in console

Fetches product: `page.tsx:71`
`{imageUrl: 'https://cdn.sanity.io/images/ghi85048/production/e_bd499c8520640d4c6cddaa853415cba7a7d4-500x500.webp', description: 'A smooth blend of vanilla ice cream, your favorite... and topped with whipped cream and oreo cookies.\n\n', price: 6, original Price: 10, available: true, ...}`

total items: 0 `CartContext.tsx:61`

Product added: Oreo Frappe, Quantity: 2 `CartContext.tsx:55`

Product added to cart `CartContext.tsx:39`

total items: 2 `CartContext.tsx:61`

Update Quantity

☐ Hide network
 ☐ Log XMLHttpRequests

☐ Preserve log
 ☒ Eager evaluation

☐ Selected context only
 ☒ Autocomplete from history

☒ Group similar messages in console
 ☒ Treat code evaluation as user action

☒ Show CORS errors in console

Fetches products: `utils.ts:16`
`(14) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}]`

total items: 0 `CartContext.tsx:61`

Product added: Oreo Frappe, Quantity: 2 `CartContext.tsx:55`

Product added to cart `CartContext.tsx:39`

total items: 2 `CartContext.tsx:61`

total items: 3 `CartContext.tsx:61`

Remove

The screenshot shows a web application interface for a shopping cart. The cart contains three items: Country Burger (\$45), Pizza (\$43), and Sandwiches (\$25). The total bill is \$161. A coupon code field is present. The Chrome DevTools console shows a warning from warn-once.js:16 about image aspect ratio, and two messages from CartContext.tsx:65 and 61.

Product	Price	Quantity	Total	Remove
Country Burger	\$45	1	\$45	x
Pizza	\$43	2	\$86	x
Sandwiches	\$25	1	\$25	x

Coupon Code

Enter Here Code

Total Bill

Cart Subtotal	156
Shipping Charges	\$05.00
Total	\$161

Console Messages:

- Warning: Image with src "https://cdn.sanity.io/images/ghi85848/..." has either width or height modified, but not the other. If you use CSS to change the size of your image, also include the styles 'width: "auto"' or 'height: "auto"' to maintain the aspect ratio.
- Product removed from cart (CartContext.tsx:65)
- total items: 4 (CartContext.tsx:61)

Logs from Testing Tools

Lighthouse

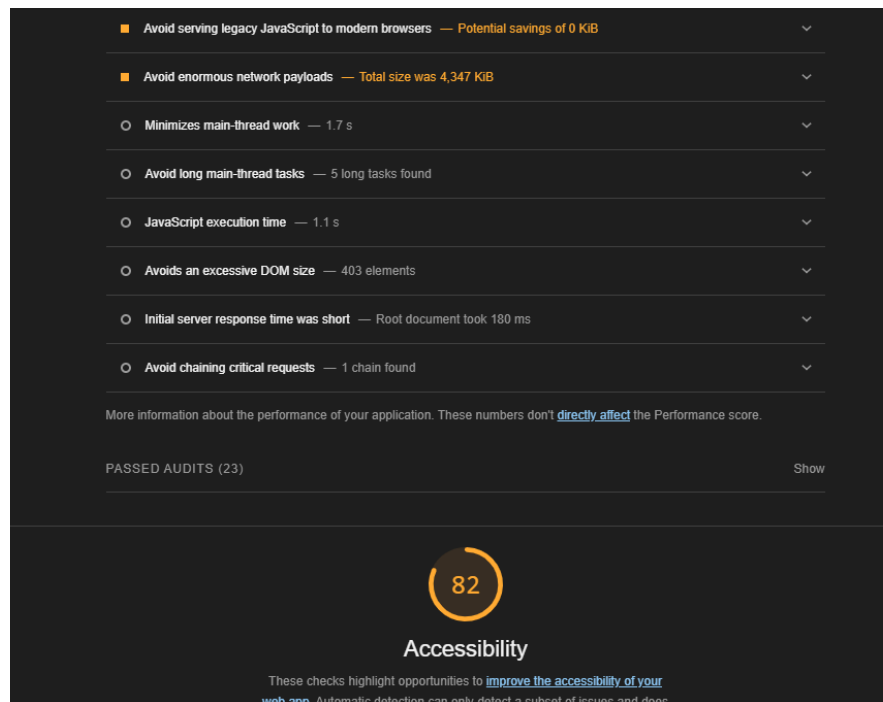
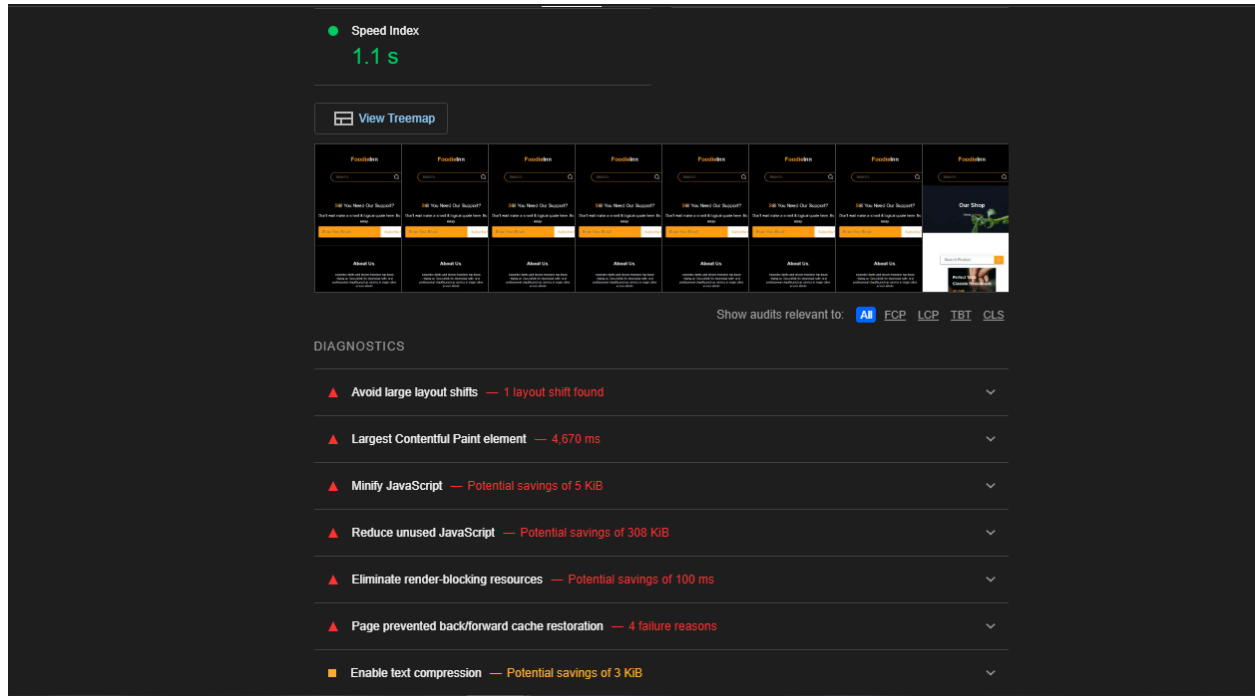
The screenshot shows the Lighthouse performance audit results for a web application. The overall performance score is 27. The audit includes metrics for First Contentful Paint (0.4 s), Largest Contentful Paint (4.7 s), Total Blocking Time (850 ms), and Cumulative Layout Shift (0.605).

Performance Score: 27

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

Metrics:

- First Contentful Paint: 0.4 s
- Largest Contentful Paint: 4.7 s
- Total Blocking Time: 850 ms
- Cumulative Layout Shift: 0.605



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

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

Save Share

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

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Body Cookies Headers (6) Test Results

200 OK · 1.00 s · 4.68 KB

JSON Preview Visualize

```
1 [
2   {
3     "name": "Oreo Frappe",
4     "imageUrl": "https://cdn.sanity.io/images/gh185048/production/ee97bd499c8520640d4c6cddaa853415c8a7a7d4-500x500.webp",
5     "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",
6     "price": 6,
7     "originalPrice": 10,
8     "category": "Drink",
9     "id": 12
10  },
11  {
12    "description": "An icy blend of pineapple and coconut",
13    "price": 8,
14    "originalPrice": null,
15    "category": "Drink",
16    "id": 14,
17    "name": "Pina Colada",
18    "imageUrl": "https://cdn.sanity.io/images/gh185048/production/bcd@fd449c87c8ea41821ac3716f32aefad2f24c-640x640.webp"
19  },
20  {
21    "id": 13,
22    "name": "Cold Coffee",
23    "imageUrl": "https://cdn.sanity.io/images/gh185048/production/6189148d8bebd33a2c2231d2fae9a78132a6a91a-500x500.webp",
24    "description": "A smooth blend of vanilla ice cream, coffee extracts, milk and sugar and topped with whipped cream, coffee and chocolate syrup",
25    "price": 9,
26    "originalPrice": 14,
27    "category": "Drink"
28  },
29 ]
```

Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

Error Handling for status 500

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

Save Share

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

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

Body Cookies Headers (6) Test Results

500 Internal Server Error · 4.20 s · 281 B

JSON Preview Visualize

```
1 {
2   "error": "Server Error. Please try again"
3 }
```

Successful fetching of individual product

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/gh185048/production/95a970acfaa0bc5e7df93be9527c2d8a1bc93562-1248x1068.png",
  "description": "Juicy beef burger with fresh lettuce, tomatoes, and cheese."
}
```

Error Handling for status 500

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

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

404 Not Found · 3.35 s · 256 B

```
{
  "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
1  import { client } from "@sanity/lib/client";
2  import { NextResponse } from "next/server";
3
4  export async function GET() {
5    try {
6
7      const query = `*[_type == "food"] {
8        id,
9        name,
10       "imageUrl": image.asset->url,
11       description,
12       price,
13       originalPrice,
14       category,
15     }`;
16
17     const products = await client.fetch(query)
18
19     if (!products){
20       return NextResponse.json({error: "Products Not Found"}, {status: 404});
21     }
22
23     return NextResponse.json(products, {status: 200});
24   } catch (error) {
25     console.error("Server Error: ", error)
26     return NextResponse.json({error: "Server Error. Please try again"}, {status: 500});
27   }
28 }
```

```
src > lib > TS utils.ts > ...
4  export function cn(...inputs: ClassValue[]) {
5      return twMerge(clsx(inputs))
6  }
7
8  export const fetchProducts = async (api: string) => {
9      try {
10         const res = await fetch(api);
11         const data = await res.json();
12
13         if(!res.ok) {
14             throw new Error(data.error || "Failed to load products");
15         }
16         console.log("Fetched products: ", data);
17         return {data};
18     } catch (error: any) {
19         console.error("Error fetching products", error);
20         return {error: error.message};
21     }
22 }
23
```

```
src > app > shop > ⚙️ page.tsx > 🏠 ShopPage
32  export default function ShopPage() {
40
41      useEffect(() => {
42          fetchProducts('/api/shop').then((res) => {
43              if (res.error) {
44                  setError(res.error);
45                  return;
46              }
47
48              if (res.data) {
49                  setMenu(res.data);
50              }
51
52              setError('');
53              setLoading(false)
54          })
55      }, []);
56
```

Fallback UI Examples

```

95      {/* Food Cards */}
96      <div className="col-span-9 row-auto flex flex-wrap md:justify-normal justify-center mt-8">
97        {error && (
98          <p className="mx-auto mt-8">{error}</p>
99        )}
100        {paginatedMenu && (
101          paginatedMenu.map((food, idx) => (
102            <Link key={idx} href={` /shop/${food.id}` } >
103              <ShopCard
104                imagePath={food.imageUrl}
105                altText={food.name}
106                imageHeight={220}
107                imageWidth={244}
108                dishName={food.name}
109                currentPrice={food.price}
110                oldPrice={food.originalPrice}
111              />
112            </Link>
113          ))
114        )}
115      </div>
116

```

```

src > app > shop > [food_id] > page.tsx > ShopDetails > fetchProduct
36  export default function ShopDetails() {
79    useEffect(() => {
100      }, [food_id]);
109
110
111    if (loading) return <div>loading...</div>;
112    // if (error) return <div>{error}</div>;
113
114    > const socialIcons = [
145      ]
146
147    const handleNext = () => {
148      if (currentIndex + itemsPerPage < allFood.length){
149        setCurrentIndex((prev) => prev + 1);
150      }
151    }
152
153    const handlePrev = () => {
154      if (currentIndex > 0) {
155        setCurrentIndex((prev) => prev - 1);
156      }
157    }
158
159
160    return (
161      <>
162        <Banner Title="Shop Details" Page="Shop details" />
163        <div className="bg-white">
164          {foodData ? (
165            <div className="container max-w-screen-lg mx-auto grid grid-cols-1 md:grid-cols-2 gap-y-4 md:gap-4 py-16 text-[#333333]">...
397          </div>
398          ) : ( <div className="flex justify-center items-center h-screen text-4xl"><p className=" " >{error}</p></div>
399          )
400        </div>
401      </>
402    )
403  }
404

```

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
1  NEXT_PUBLIC_SANITY_PROJECT_ID=
2  NEXT_PUBLIC_SANITY_DATASET=
3  NEXT_PUBLIC_SANITY_API_TOKEN=
```

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	<input checked="" type="checkbox"/>
Performance Testing	<input checked="" type="checkbox"/>
Error Handling	<input checked="" type="checkbox"/>
Device Testing	<input checked="" type="checkbox"/>
Security Testing	<input checked="" type="checkbox"/>
Documentation	<input checked="" type="checkbox"/>
