

## Day 5 - Testing and Backend Refinement - [SHOP.CO]

### 1. Introduction

This report outlines the testing activities conducted for the E-Commerce Marketplace project. The primary objectives of the testing were to ensure functionality, performance, error handling, and cross-browser compatibility, preparing the application for deployment.

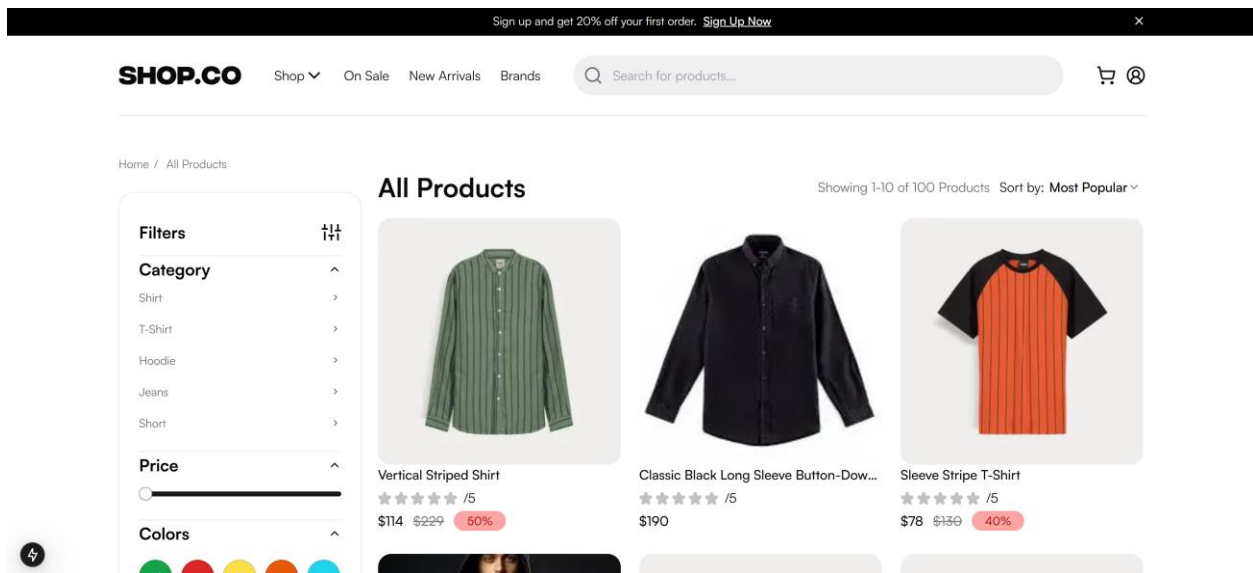
The testing was conducted using a combination of manual and automated tools such as **Postman**, **Lighthouse**, and **BrowserStack** etc. The findings and recommendations are detailed below.

### 2. Testing Overview

#### Types of Testing Performed

- **Functional Testing:** Validating core features such as product listing, cart functionality, and product detailed pages.

#### Products Listing Page:



#### Product Detailed Page:

Home / Shop / Men / T-shirts



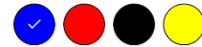
## CASUAL GREEN BOMBER

★★★★★ /5

**\$240** ~~\$300~~ 20%

This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The minimalist style makes it perfect for layering over casual t-shirts or hoodies. Whether you're out with friends or just lounging, this jacket provides a laid-back yet fashionable vibe. Its muted green color adds a subtle, earthy tone that pairs well with a variety of outfits, making it a versatile addition to your casual wardrobe.

Select Colors:



Choose Size



Add to Cart

### Cart Functionality:

**SHOP.CO**

Shop ▾

On Sale

New Arrivals

Brands

Q shirt



Home / Cart

## YOUR CART



### Checkered Shirt

Size: XL

Color: White

**\$178**

- 1 +



### Classic Black Long Sleeve Button-Down Shirt

Size: S

Color: Yellow

**\$190**

- 1 +



### Order Summary

Subtotal **\$368**

Discount (-20%) **-\$74**

Delivery Fee **\$15**

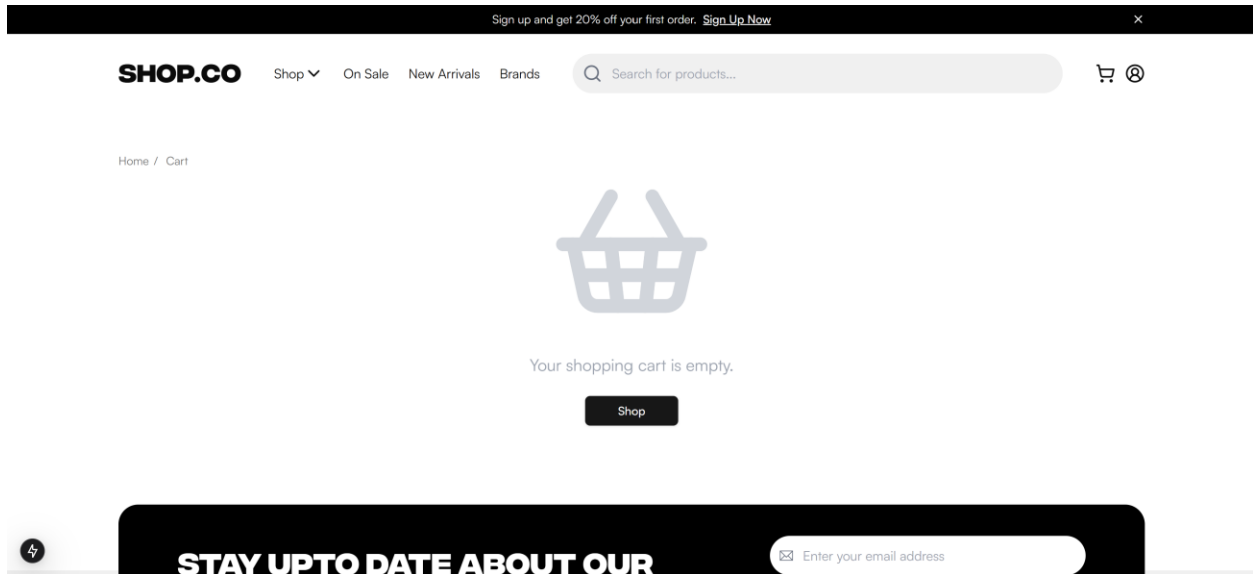
Total **\$279**

Add promo code

Apply

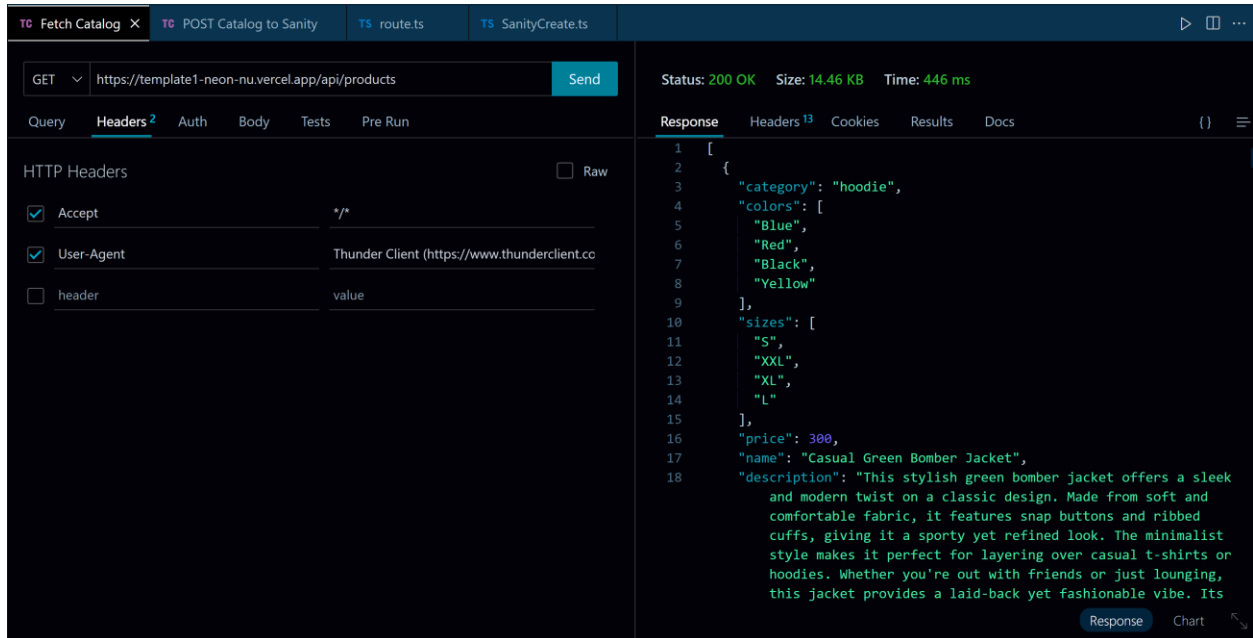
Go to Checkout →

### FallBack UI when Cart is Empty:



- **API Testing:** Ensuring backend endpoints respond correctly and error handling works as expected.

Data Fetching using GET:



Fetches Data to /api/catalog to upload on sanity.

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/catalog`. The request body is a JSON object representing a catalog item. The response is a 200 OK status with a 943 Byte size and 327 ms time. The response body is a JSON object with a success message and the catalog data.

```
POST http://localhost:3000/api/catalog Send
```

Query Headers<sup>2</sup> Auth **Body<sup>1</sup>** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 [
2   {
3     "category": "hoodie",
4     "colors": [
5       "Blue",
6       "Red",
7       "Black",
8       "Yellow"
9     ],
10    "sizes": [
11      "S",
12      "XXL",
13      "XL",
14      "L"
15    ],
16    "price": 300,
17    "name": "Casual Green Bomber Jacket",
18    "description": "This stylish green bomber jacket offers a sleek and comfortable fabric, it features snap buttons and
```

Status: 200 OK Size: 943 Bytes Time: 327 ms

Response Headers<sup>5</sup> Cookies Results Docs {} ≡

```
1 {
2   "message": "Catalog created successfully",
3   "data": [
4     {
5       "category": "hoodie",
6       "colors": [
7         "Blue",
8         "Red",
9         "Black",
10        "Yellow"
11      ],
12      "sizes": [
13        "S",
14        "XXL",
15        "XL",
16        "L"
17      ],
18      "price": 300,
19      "name": "Casual Green Bomber Jacket",
20      "description": "This stylish green bomber jacket offers a sleek and modern twist on a classic design. Made from soft and comfortable fabric, it features snap buttons and ribbed cuffs, giving it a sporty yet refined look. The minimalist style makes it perfect for layering over casual t-shirts or
```

Response Chart ↗

API Error Handling:

The screenshot shows a REST client interface with a POST request to `http://localhost:3000/api/catalog`. The response is a 500 Internal Server Error status with a 78 Byte size and 226 ms time. The response body is a JSON object with an error message.

```
POST http://localhost:3000/api/catalog Send
```

Query Headers<sup>2</sup> Auth **Body** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1
```

Status: 500 Internal Server Error Size: 78 Bytes Time: 226 ms

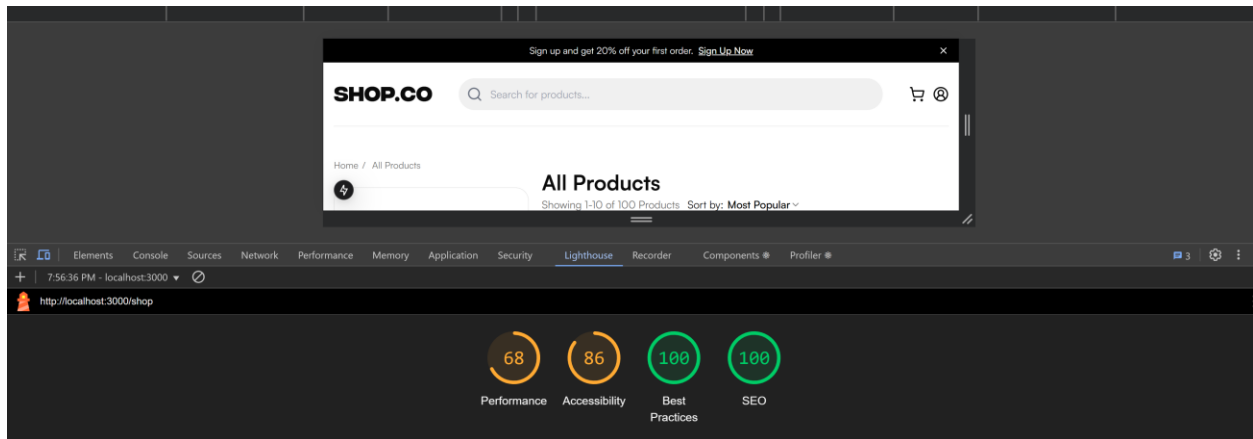
Response Headers<sup>5</sup> Cookies Results Docs {} ≡

```
1 {
2   "error": "Failed to create catalog SyntaxError:
3     Unexpected end of JSON input"
}
```

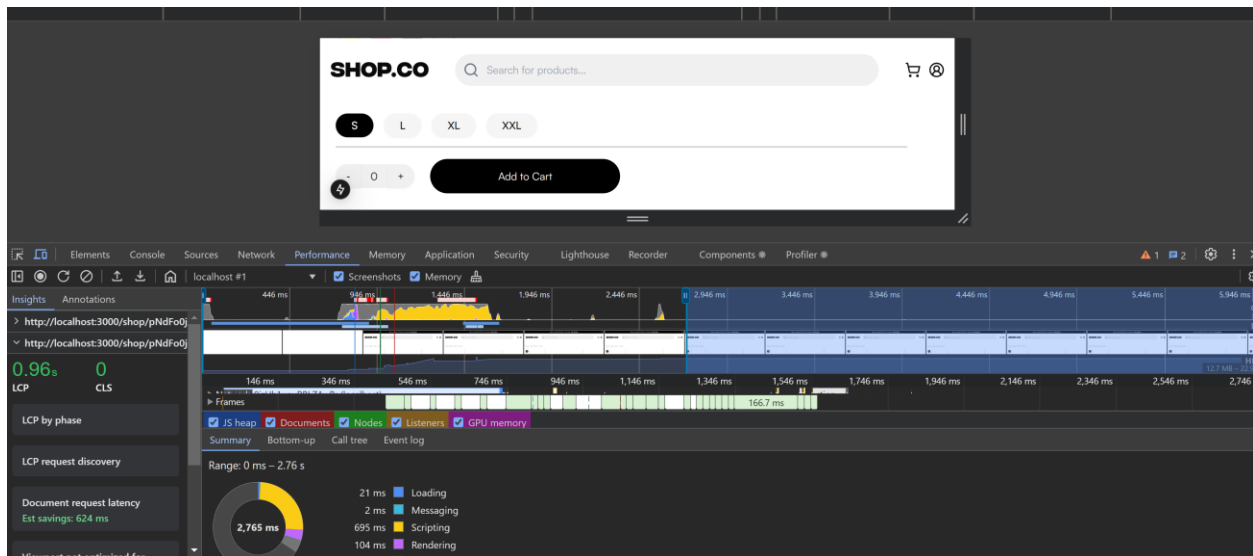
Response Chart ↗

- **Performance Testing:** Measuring page load times and optimizing asset delivery.

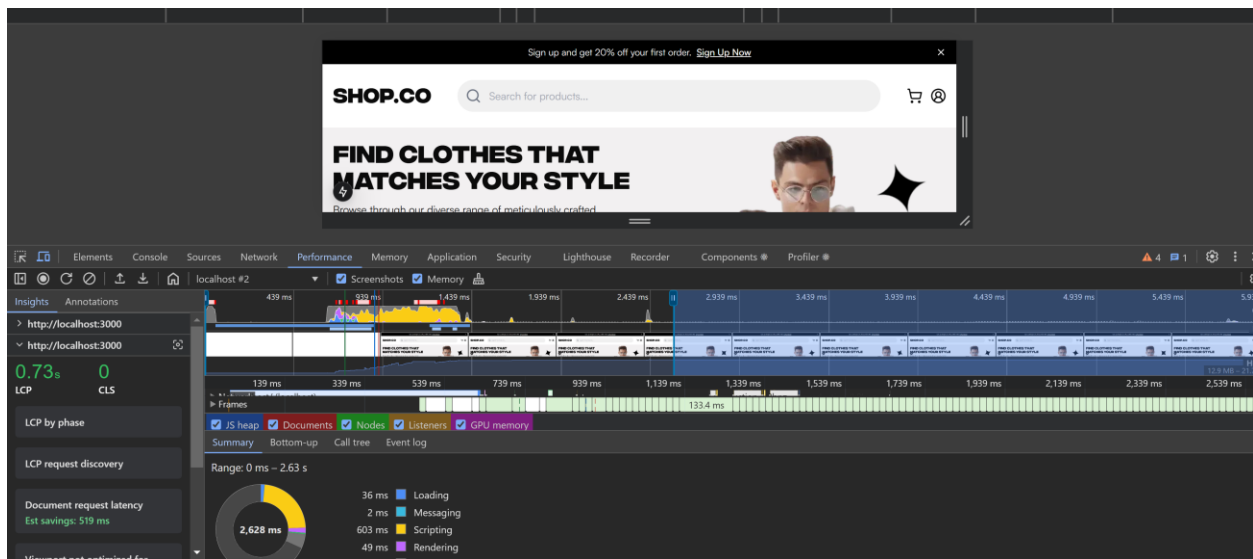
## Using Browser Dev Tool:



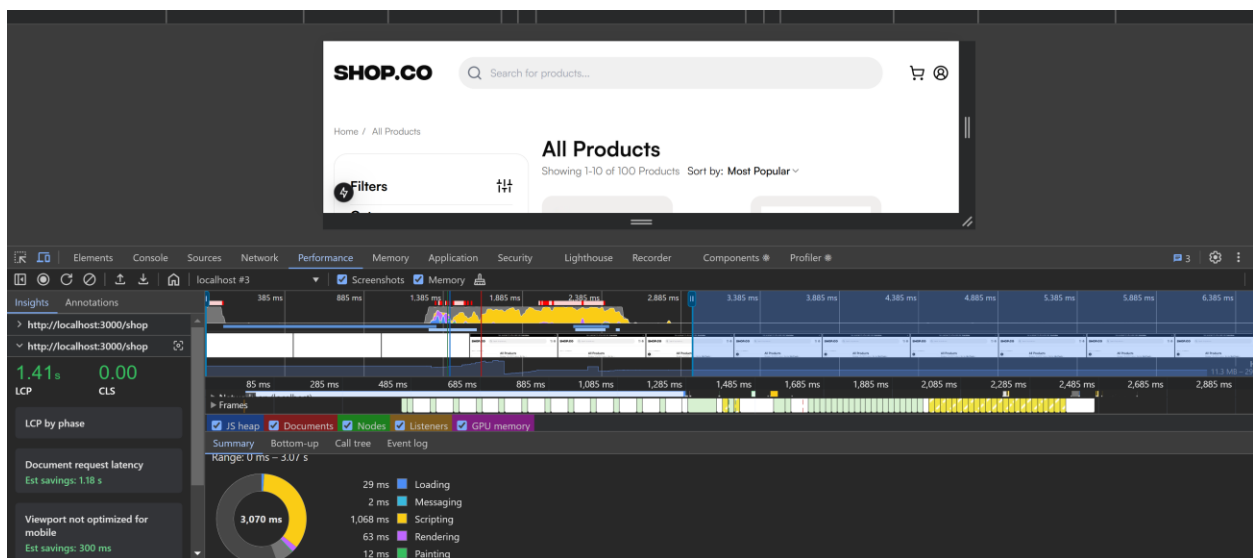
## Add To Cart:




## Home Page:



## Product Listing Page:




## Using GTMetrix:




## Latest Performance Report for:

<https://shopco-uiux-hackathon.vercel.app/>

Report generated: Mon, Jan 20, 2025 7:27 AM -0800

Test Server Location:  Vancouver, Canada

Using:  Chrome 117.0.0.0, Lighthouse 11.0.0

### GTmetrix Grade ?

<b>A</b>	Performance ? <b>99%</b>	Structure ? <b>94%</b>
----------	-----------------------------	---------------------------

### Web Vitals ?

LCP ? <b>851ms</b>	TBT ? <b>0ms</b>	CLS ? <b>0.06</b>
-----------------------	---------------------	----------------------

## Using Lighthouse Metrix:

URL  
<https://shopco-uiux-hackathon.vercel.app/>

State  
● succeeded







Device  
Mobile

Version  
12.2.1

Request Time  
4 minutes ago







### Performance Scores

View Lighthouse Performance scores across all regions

 <b>US West</b> us-west1 <b>88</b> / 100 FCP: 989ms LCP: 2.3s TBT: 402ms	 <b>US East</b> us-east4 <b>99</b> / 100 FCP: 998ms LCP: 1.8s TBT: 119ms	 <b>Finland</b> europe-north1 <b>96</b> / 100 FCP: 1.2s LCP: 2.4s TBT: 160ms
 <b>Germany</b> europe-west3 <b>96</b> / 100 FCP: 1.1s LCP: 2.3s TBT: 158ms	 <b>Japan</b> asia-northeast1 <b>99</b> / 100 FCP: 1.2s LCP: 1.5s TBT: 73ms	 <b>Australia</b> australia-southeast1 <b>98</b> / 100 FCP: 1.3s LCP: 1.3s TBT: 119ms

### Performance Metrics

Inspect each metric that leads to the performance score

						
First Contentful Paint <a href="#">↗</a>	989ms	998ms	1.2s	1.1s	1.2s	1.3s
Speed Index <a href="#">↗</a>	1.5s	1.1s	2.3s	1.9s	2.2s	2.9s
Largest Contentful Paint <a href="#">↗</a>	2.3s	1.8s	2.4s	2.3s	1.5s	1.3s
Total Blocking Time <a href="#">↗</a>	402ms	119ms	160ms	158ms	73ms	119ms
Cumulative Layout Shift <a href="#">↗</a>	0	0	0	0	0	0

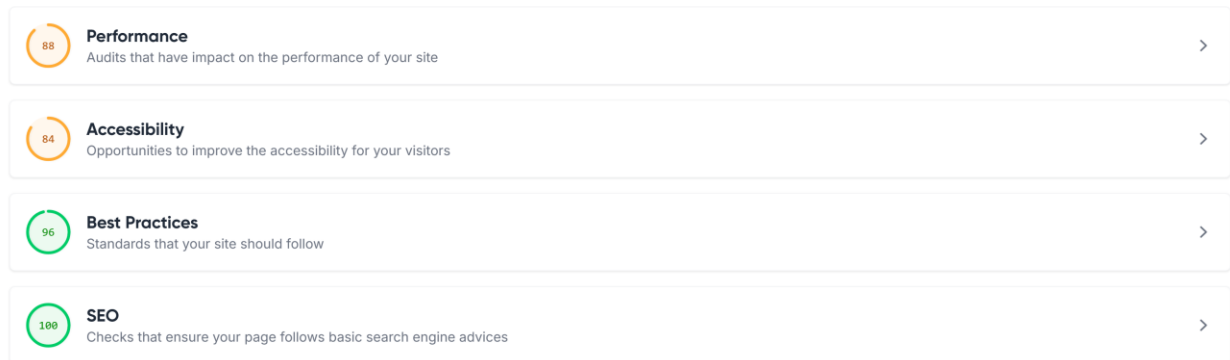
## Transferred Assets

Review transferred assets and their size



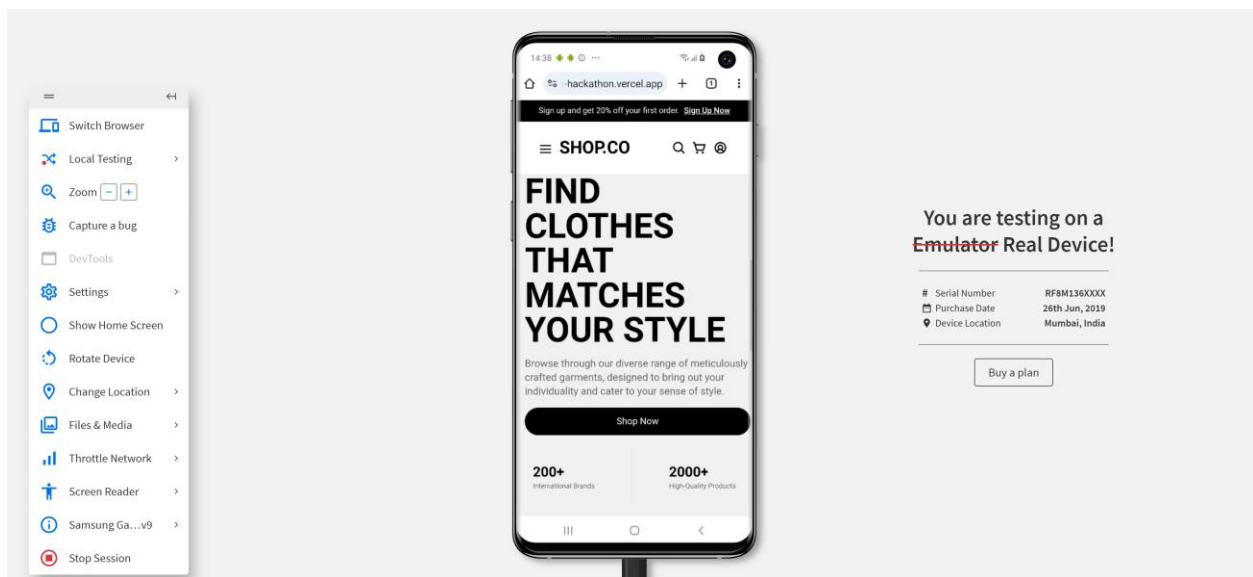
## Lighthouse Audits

Review passed and failed audits for each Lighthouse score



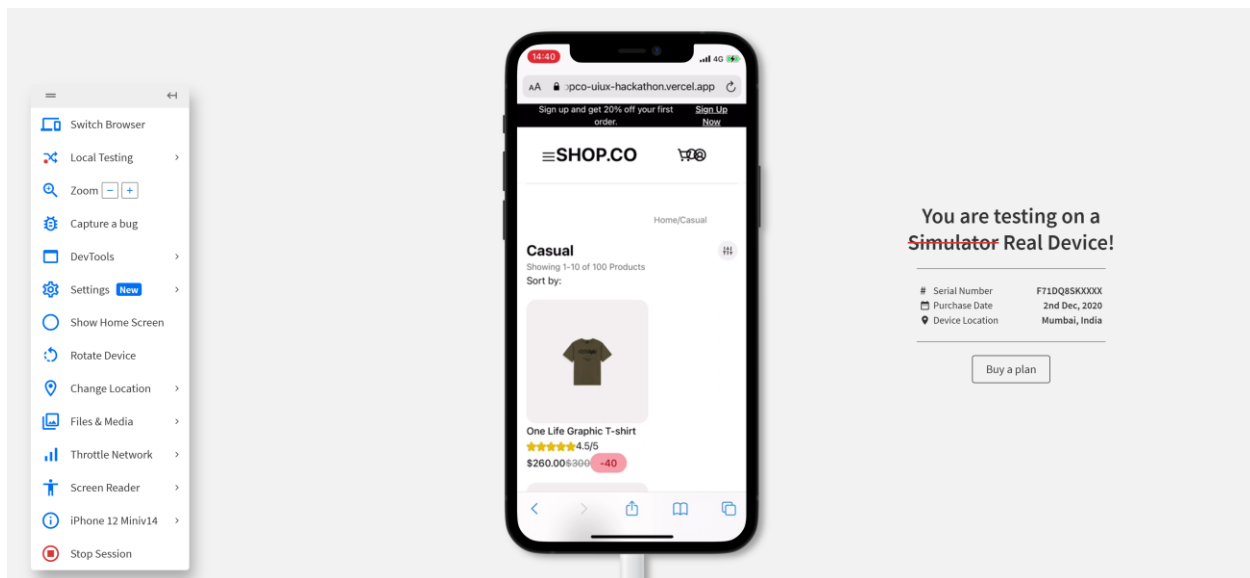
- **Cross-Browser and Device Testing:** Ensuring responsiveness and compatibility on various browsers and devices.

Chrome in Samsung Galaxy-S10:

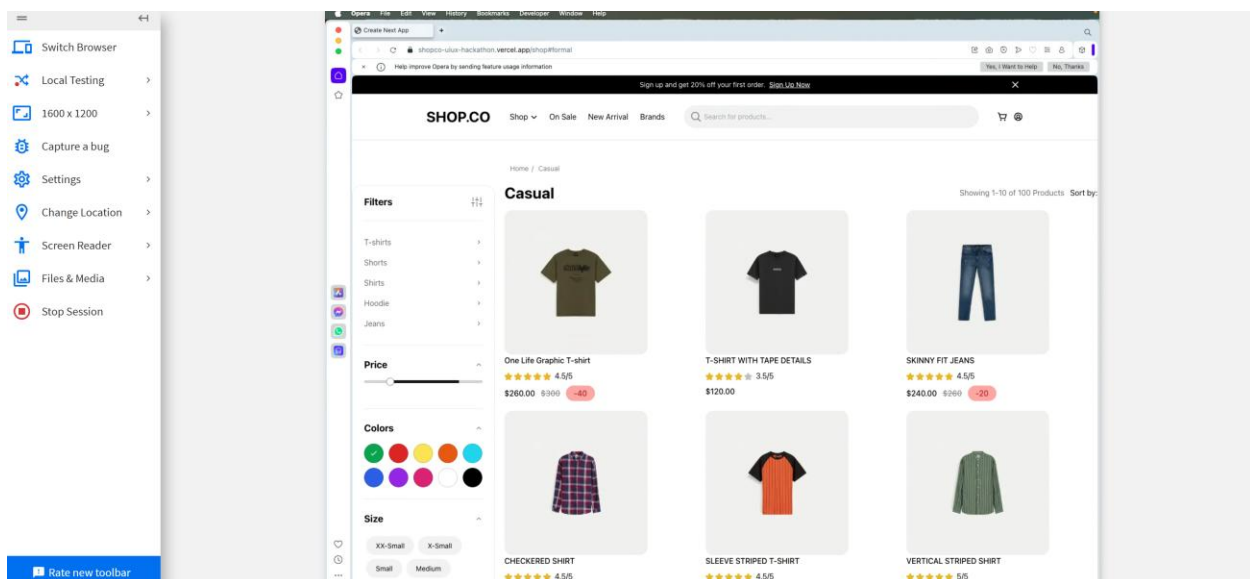


Firefox in Iphone-12:





Opera in 1600 x 1200 Tablet:



- **Error Handling Testing:** Verifying those fallback mechanisms provide a seamless user experience.

Error Handling using Try-catch:

```

export async function POST(request: NextRequest) {
  try {
    const data = await request.json()
    await data.map((product: ProductType) => {
      return client.create({
        _type: 'catalog',
        name: product.name,
        price: product.price,
        category: product.category,
        discountPercent: product.discountPercent,
        isNew: product.isNew,
        colors: product.colors,
        sizes: product.sizes,
        description: product.description,
        imageUrl: product.imageUrl ? {
          _type: 'image',
          asset: {
            _type: 'reference',
            _ref: product.imageUrl
          },
        } : null,
      })
    })
    return NextResponse.json({ message: 'Catalog created successfully', data });
  } catch (error) {
    return NextResponse.json({ error: `Failed to create catalog ${error}` }, { status: 500 });
  }
}

```

```

try {
  const filePath = path.join(process.cwd(), 'public', 'item', `${imageName}`);
  const imageFile = createReadStream(filePath);
  const imageAsset = await client.assets.upload("image", imageFile, { filename: basename(filePath) })
  console.log(`Image Uploaded Successfully: ${imageAsset._id}`);
  return imageAsset._id
} catch (error) {
  return `Failed to upload image, ${error}`
}

```

```

const response = await fetch("https://template1-neon-nu.vercel.app/api/products")
data = await response.json()
data = await Promise.all(data.map(async (productData: ProductType) => {
  try {
    const imageResponse = await fetch(productData.imageUrl);

    if (!imageResponse.ok) {
      throw new Error(`Failed to fetch image: ${productData.imageUrl}`);
    }

    const arrayBuffer = await imageResponse.arrayBuffer();
    const imageBuffer = Buffer.from(arrayBuffer);
    const imageAsset = await client.assets.upload("image", imageBuffer);

    return {
      ...productData,
      imageUrl: imageAsset._id
    };
  } catch (error) {
    console.error(`Error processing product ${productData.name}:`, error);
    return productData;
  }
})

```

Fallbacks In case no product is available to Fetch:

```

<Container className="relative overflow-x-hidden">
  {catalog.length > 0 ? (
    catalog.map(
      (product) =>
        product._id === id && (
          <div key={product._id} className="flex flex-col">
            <Breadcrumb location={["Home", "Shop", "Men", "T-shirts"]} />
            <Product product={product} />
          </div>
        )
    )
  ) : (
    <p>No products available.</p>
  )}
  <FeedbackTabs />
  <DisplayCard heading="You might also like" button={false} id={id} />
</Container>

```

## 2. Observations:

Performance on lower side mainly because of images not optimized can be further improved by optimizing image delivery and enabling caching.

Compress large images and enable lazy loading for non-critical assets.