

Day 5 - Testing and Backend Refinement of E - Commerce Website

As we progress toward launching our marketplace, Day 5 is dedicated to ensuring that all components are fully tested, optimized for performance, and ready to handle real-world customer interactions.

Key Areas of Focus :-

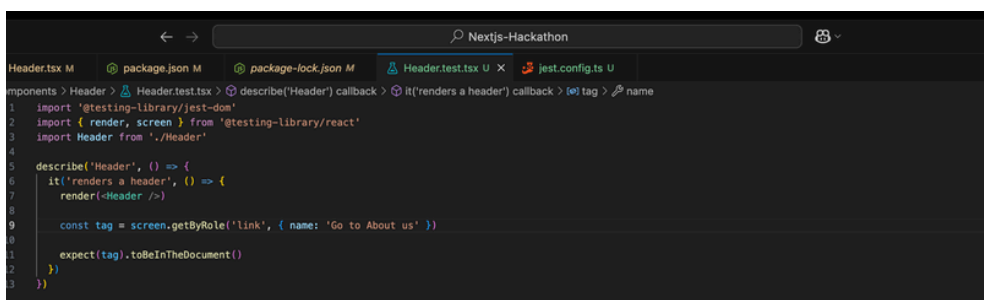
1. Functional Testing :

This document outlines the functional testing strategy for the Next.js e-commerce application. The testing covers core features such as product listing, filtering, cart operations, and dynamic routing using automated and manual testing approaches.

Tested Core Features :

1.1 Client-Side Navigation Between Components :

- **Tested with React Testing Library:**
- Verified smooth navigation across different sections of the website (e.g., home page to product listing, product details, cart, and wishlist pages)
- Ensured navigation links and buttons function correctly without a full page reload



```
Header.test.tsx M package.json M package-lock.json M Header.test.tsx U X jest.config.ts U
components > Header > Header.test.tsx > describe('Header') callback > it('renders a header') callback > tag > name
1 import '@testing-library/jest-dom'
2 import { render, screen } from '@testing-library/react'
3 import Header from './Header'
4
5 describe('Header', () => {
6   it('renders a header', () => {
7     render(<Header />)
8
9     const tag = screen.getByRole('link', { name: 'Go to About us' })
10
11     expect(tag).toBeInTheDocument()
12   })
13 })
```

```
PASS components/Header/Header.test.tsx
Header
  ✓ renders a header (52 ms)

Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        0.495 s, estimated 1 s
Ran all test suites.
muhammadasan~$ ~/Developer/Nextjs/Nextjs-Hackathon
```

1.2 Cart operations :

1.2.1 Add To Cart functionality :

- **Tested with React Testing Library:**
- Ensured that clicking the "Add to Cart" button correctly adds the selected product to the cart. Verified that the cart reflects the added item with accurate details (name, price, and quantity).

```
components > ProductsAdder > ProductsAdder.test.tsx > ...
1  import '@testing-library/jest-dom';
2  import { render, screen } from '@testing-library/react';
3  import ProductsAdder from './ProductsAdder'; // Update the path
4  import useCart from '../context/CartContextProvider'
5
6  // Mock the custom hook
7  jest.mock('../context/CartContextProvider', () => ({
8    __esModule: true,
9    default: jest.fn(),
10 }));
11
12 describe('Add to Cart Button in ProductsAdder', () => {
13   const mockSetIncrement = jest.fn();
14
15   beforeEach(() => {
16     // Reset all mock implementations before each test
17     jest.clearAllMocks();
18
19     // Mock the hook with its implementation
20     (useCart as jest.Mock).mockReturnValue({
21       setIncrement: mockSetIncrement,
22       cartItems: {},
23     });
24   });
25
26   it('renders the Add to Cart button', () => {
27     render(
28       <ProductsAdder
29         stock={10}
30         Id={1}
31         name="Test Product"
32         price={100}
33         image="test-image.jpg"
34         colors="red"
35       />
36     );
37
38     const button = screen.getByRole('button', { name: 'ADD TO CART' });
39     expect(button).toBeInTheDocument();
40   });
41
42
43 });
44
```

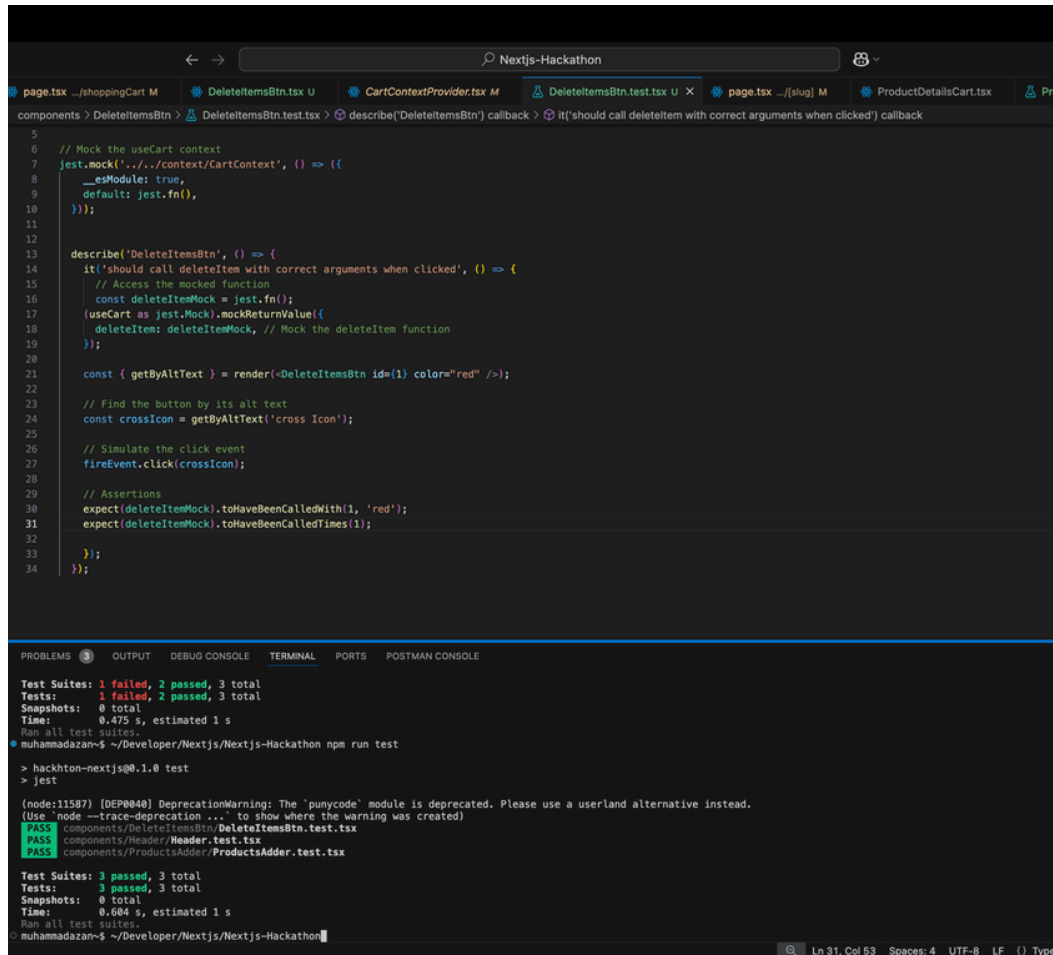
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```
(Use `node --trace-deprecation ...` to show where the warning was created)
PASS components/Header/Header.test.tsx
PASS components/ProductsAdder/ProductsAdder.test.tsx

Test Suites: 2 passed, 2 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.43 s, estimated 1 s
Ran all test suites.
muhammadasan~$ ~/Developer/Nextjs/Nextjs-Hackathon
```

1.2.2 Remove From Card functionality :

- **Tested with React Testing Library:**
- Confirmed that clicking the "Remove from Card" button successfully removes the item from the cart, updating the cart view accordingly



```
5
6 // Mock the useCart context
7 jest.mock('../context/CartContext', () => ({
8   __esModule: true,
9   default: jest.fn(),
10 }));
11
12
13 describe('DeleteItemBtn', () => {
14   it('should call deleteItem with correct arguments when clicked', () => {
15     // Access the mocked function
16     const deleteItemMock = jest.fn();
17     (useCart as jest.Mock).mockReturnValue({
18       deleteItem: deleteItemMock, // Mock the deleteItem function
19     });
20
21     const { getByAltText } = render(<DeleteItemBtn id={1} color="red" />);
22
23     // Find the button by its alt text
24     const crossIcon = getByAltText('cross icon');
25
26     // Simulate the click event
27     fireEvent.click(crossIcon);
28
29     // Assertions
30     expect(deleteItemMock).toHaveBeenCalledWith(1, 'red');
31     expect(deleteItemMock).toHaveBeenCalledTimes(1);
32
33   });
34 });
```

Test Suites: 1 failed, 2 passed, 3 total
Tests: 1 failed, 2 passed, 3 total
Snapshots: 0 total
Time: 0.475 s, estimated 1 s
Run all test suites.

muhammadasan-@Developer/Nextjs/Nextjs-Hackathon npm run test

```
> hackton-nextjs@0.1.0 test
> jest

(node:11587) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS components/DeleteItemBtn/DeleteItemBtn.test.tsx
PASS components/Header/Header.test.tsx
PASS components/ProductsAdder/ProductsAdder.test.tsx

Test Suites: 3 passed, 3 total
Tests: 3 passed, 3 total
Snapshots: 0 total
Time: 0.064 s, estimated 1 s
Run all test suites.
```

1.2.3 Cart updates correctly upon item addition/removal :

Tested real-time updates to the cart when items are added or removed, ensuring accurate total price and quantity adjustments.

1.3 WishList operation :

1.3.1 Add To WishList functionality :

- **Tested with React Testing Library:**

- Verified that clicking the "Add to Wishlist" button correctly adds the selected product to the wishlist. Ensured that the product appears in the wishlist with accurate details.

```

components > Add-To-Wishlist-Icons_functionality > WishlistIcon > WishlistIcon.test.tsx U x
1 import '@testing-library/jest-dom';
2 import { render, screen } from '@testing-library/react';
3 import HeartIcon from './WishlistIcon'; // Update the path
4 import useWishlist from '../../context/WishlistContextProvider'
5
6 // Mock the custom hook
7 jest.mock('../../context/WishlistContextProvider', () => ({
8   __esModule: true,
9   default: jest.fn(),
10 }));
11
12 describe('Add to Cart Button in ProductsAdder', () => {
13   const mockSetIncrement = jest.fn();
14
15   beforeEach(() => {
16     // Reset all mock implementations before each test
17     jest.clearAllMocks();
18
19     // Mock the hook with its implementation
20     (useWishlist as jest.Mock).mockReturnValue({
21       setIncrement: mockSetIncrement,
22       cartItems: [],
23     });
24   });
25
26   it('renders the Add to Cart button', () => {
27     render(
28       <HeartIcon
29         stock={10}
30         id={1}
31         name="Test Product"
32         price={100}
33         image="test-image.jpg"
34         slug="Cantilever chair"
35         color="red"/>
36     );
37
38     const heartIcon = screen.getByTestId('wishlist-icon'); // Use a test ID
39     expect(heartIcon).toBeInTheDocument();
40
41   });
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE

```

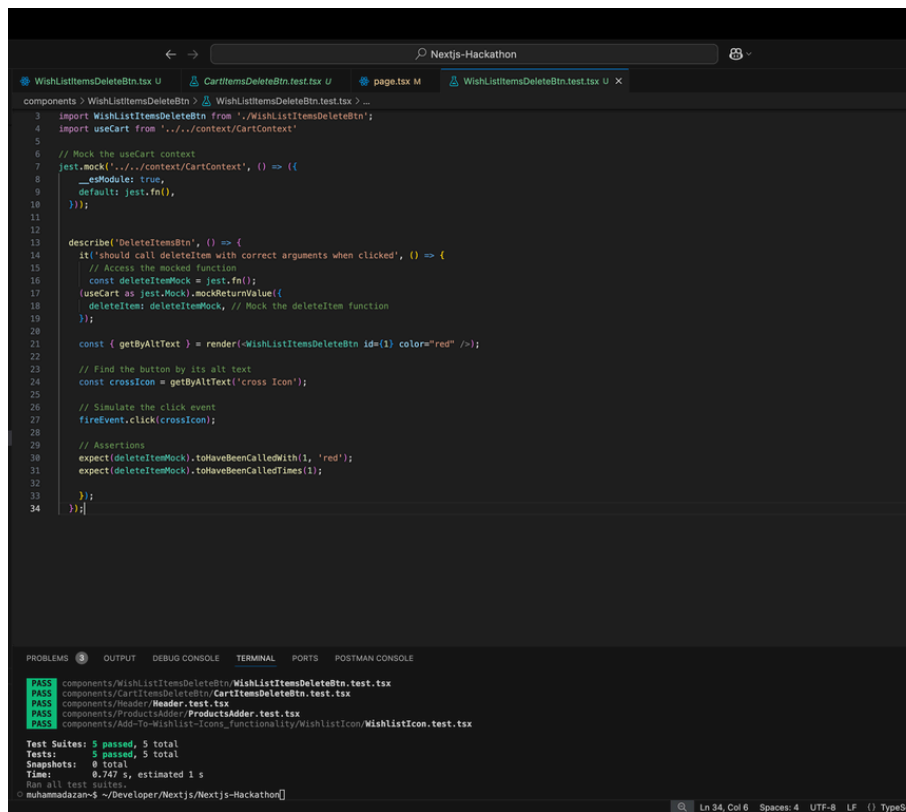
(node:17173) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.
(Use 'node --trace-deprecation ...' to show where the warning was created)
PASS components/Header/Header.test.tsx
PASS components/ProductsAdder/ProductsAdder.test.tsx
PASS components/Add-To-Wishlist-Icons_functionality/WishlistIcon/WishlistIcon.test.tsx
PASS components/DeleteItemsBtn/DeleteItemsBtn.test.tsx

Test Suites: 4 passed, 4 total
Tests: 4 passed, 4 total
Snapshots: 0 total
Time: 0.501 s, estimated 1 s
Ran all test suites.
muhammadazam$ ~/Developer/Nextjs/Nextjs-Hackathon

```

1.3.2 Remove from wishlist functionality :

- **Tested with React Testing Library:**
- Confirmed that clicking the "Remove from Wishlist" button successfully removes the item from the wishlist. Verified that the wishlist updates dynamically after removal.



```
3 import WishListItemsDeleteBtn from './WishListItemsDeleteBtn';
4 import useCart from '../context/CartContext';
5
6 // Mock the useCart context
7 jest.mock('../context/CartContext', () => ({
8   __esModule: true,
9   default: jest.fn(),
10 }));
11
12 describe('DeleteItemsBtn', () => {
13   it('should call deleteItem with correct arguments when clicked', () => {
14     // Access the mocked function
15     const deleteItemMock = jest.fn();
16     (useCart as jest.Mock).mockReturnValue({
17       deleteItem: deleteItemMock, // Mock the deleteItem function
18     });
19
20     const { getByAltText } = render(<WishListItemsDeleteBtn id={1} color="red" />);
21
22     // Find the button by its alt text
23     const crossIcon = getByAltText('cross icon');
24
25     // Simulate the click event
26     fireEvent.click(crossIcon);
27
28     // Assertions
29     expect(deleteItemMock).toHaveBeenCalledWith(1, 'red');
30     expect(deleteItemMock).toHaveBeenCalledTimes(1);
31   });
32 });
```

Test Suites: 5 passed, 5 total
Tests: 5 passed, 5 total
Snapshots: 0 total
Time: 0.747 s, estimated 1 s
Run all test suites

1.4 Product List :

- Verified that all products are displayed correctly with appropriate details (name, price, image, and description).
- Conducted manual testing to ensure proper rendering of the product listing page.

- Navigation to individual product detail pages
- Ensured product details (name, image, price, and description) are displayed correctly

1.6 Testing tools used :

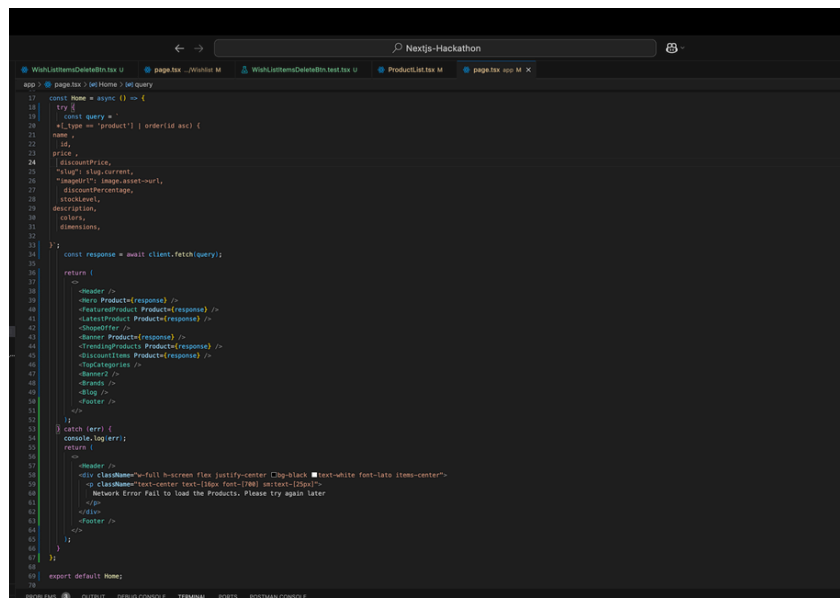
Feature	Tool Used
API Testing	Postman
Component Testing	React Testing Library
Manual Testing	Browser Navigation

Summary of Test Execution

Feature	Test Method
Product Listing Page	Manual Testing
Product Details Page	Manual Testing
Add to Cart Button	React Testing Library
Remove from Cart Button	React Testing Library
Add to Wishlist Button	React Testing Library
Remove from Wishlist Button	React Testing Library
Navigation (Dynamic Pages)	Manual Testing
Client-Side Navigation	React Testing Library

2. Add Error Handling :

- implementing the error handling and fallback UI, the component will either render the fetched product data or show an error message if the fetch fails.
- The structure of the component includes:
- **Header:** Displays the header section.
- **Product Components:** Renders components such as Hero, FeaturedProduct, LatestProduct, etc., passing the fetched response data as props.
- **Footer:** Displays the footer of the page.



```
17 const Home = async () => {
18   try {
19     const query = `
20       {
21         _type == 'product' { orderId asc {
22           name,
23           id,
24           price,
25           slug,
26           "slug": slug.current,
27           "image": image.asset.url,
28           discountPercentage,
29           stockLevel,
30           description,
31           category,
32           dimensions,
33         }
34       }
35     `
36     const response = await client.fetch(query);
37     return (
38       <>
39         <Header />
40         <Hero Product={response} />
41         <FeaturedProduct Product={response} />
42         <LatestProduct Product={response} />
43         <ShowOffer />
44         <Banner Product={response} />
45         <rendingProducts Product={response} />
46         <DiscountItems Product={response} />
47         <TopCategories />
48         <Banner />
49         <Brands />
50         <Footer />
51       </>
52     )
53   } catch (err) {
54     console.log(err);
55     return (
56       <>
57         <Header />
58         <div className="full h-screen flex justify-center flex-direction column align-items-center">
59           <div className="text-center text-[16px font-size:16px] text-[#f00]>
60             Network Error Fail to load the Products. Please try again later
61           </div>
62         </div>
63         <Footer />
64       </>
65     )
66   }
67 }
68 export default Home;
```

3. Performance Optimization :

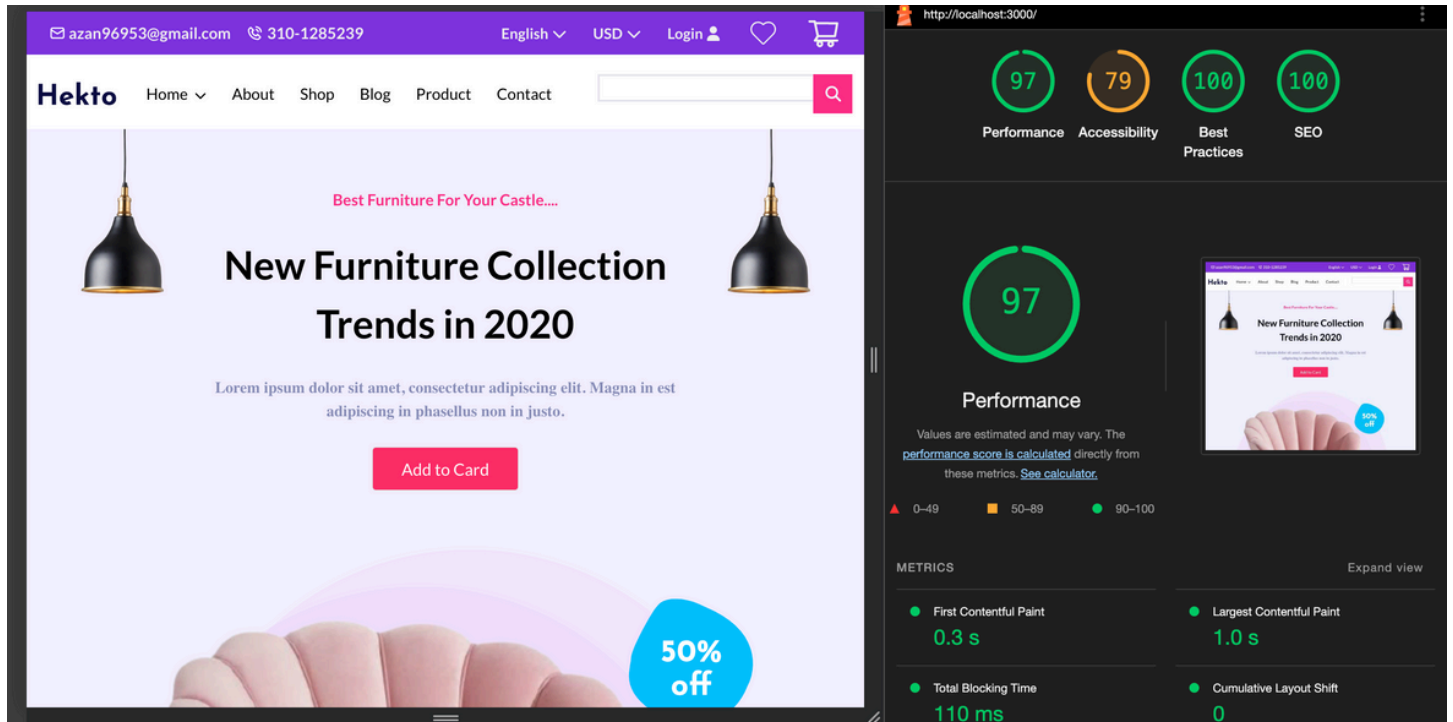
3.1. Optimize Assets

- **Image Compression:** Used tools like **TinyPNG** and **ImageOptim** to reduce image size without quality loss.
- **Lazy Loading:** Implemented **lazy loading** for large images and assets to improve load times and reduce initial page weight.
- **Eager Loading:** Applied **loading="eager"** where necessary to prioritize critical images for faster rendering.

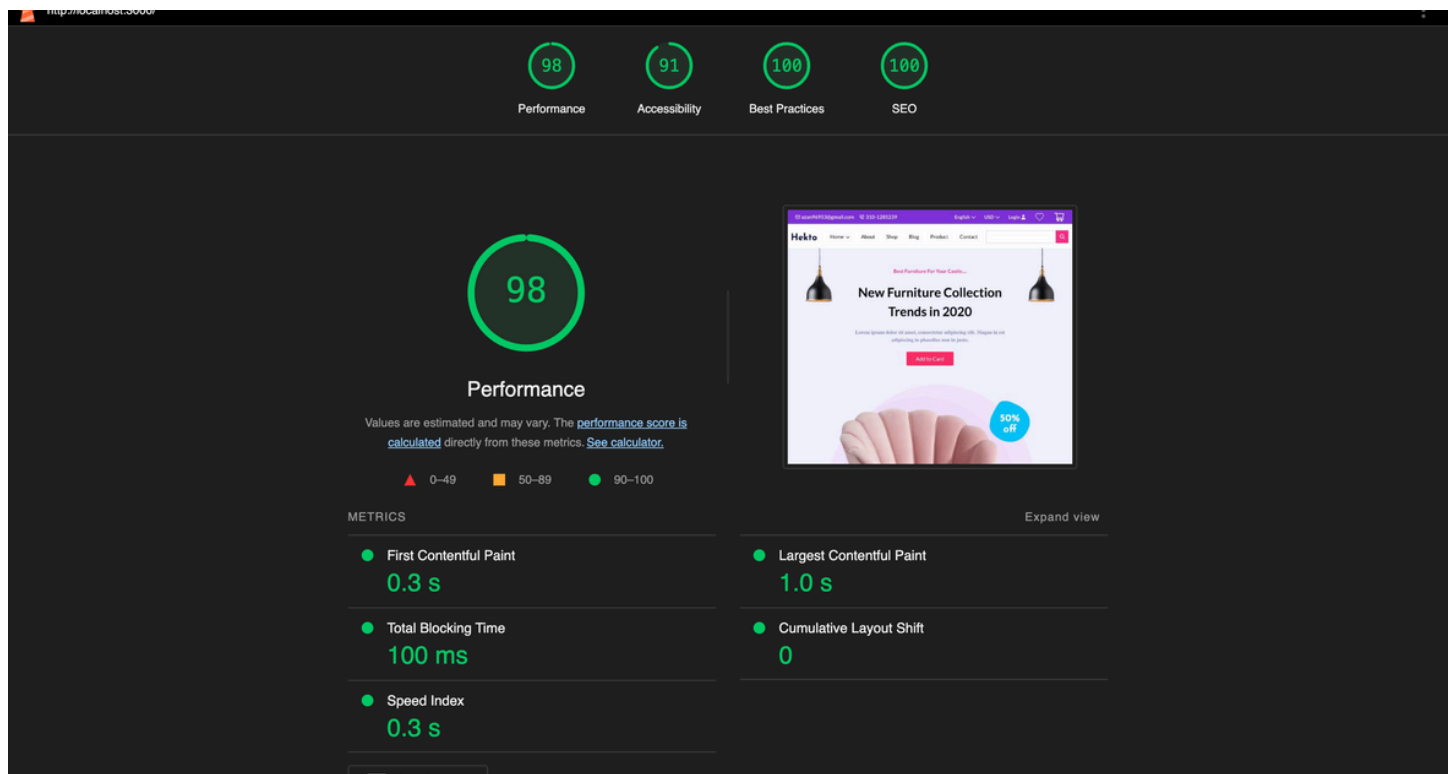
3.2. Analyze Performance

- **Lighthouse Audit:** Conducted a **performance test** using Lighthouse to identify speed and optimization issues.
- **Implemented Fixes:**
 - **Reduced unused CSS** to minimize render-blocking resources.
 - **Enabled browser caching** to store frequently accessed files locally and enhance repeat visits.
- **Optimized Painful Images:** Applied **loading="eager"** to critical images and reduced their size to enhance rendering performance.

3.3. Performance Scores (Before Optimization) :



3.4. Performance Scores (After Optimization) :



Metric	Score
Performance	98
Accessibility	91
Best Practices	100
SEO	100

4. Cross-Browser and Device Testing :

4.1. Browser Testing

- Tested on **Chrome, Firefox, Safari, and Edge** to ensure consistent rendering and functionality.

4.2. Device Testing

- Used **Website Viewer** simulate different devices.
- Manually tested on a **physical mobile device** by connecting through **localhost via network IP address**.

[illegible]