Day 5 Hackathon Report: Testing And Backend Refinement 🚀

1. *Functional Deliverables:*

Lighthouse Performance Report

1. Overall Scores:

- **Performance: 67**
- **Accessibility: 85**
- **Best Practices: 100**
- **SEO: 100**

2. Key Findings and Recommendations:

Performance (67) - Needs Improvement

● Possible Issues:

- Slow loading times.
- Inefficient resource loading.
- Render-blocking scripts or styles.

✓ Recommendations:

- Optimize images using modern formats (WebP, AVIF).
- Minify and compress CSS, JavaScript, and HTML.
- Implement lazy loading for images and iframes.
- Reduce unused JavaScript and CSS.
- Enable server-side caching and CDN usage.

Accessibility (85) - Good, but Can Improve

□ **Potential Issues:**

- Missing or low-contrast text.
- Unlabeled elements that may affect screen readers.

✅ **Recommendations:**

- Improve color contrast for better readability.
- Ensure all interactive elements have proper labels.
- Add ARIA attributes where necessary.

Best Practices (100) - Excellent

✅ **Strengths:**

- Secure implementation.
- No XSS vulnerabilities detected.
- Proper use of HTTPS and modern web standards.

⚠ **Minor Consideration:**

- Regularly update dependencies to maintain security.

SEO (100) - Excellent

✅ **Strengths:**

- Good search engine optimization.
- Proper metadata, structured data, and mobile-friendly layout.

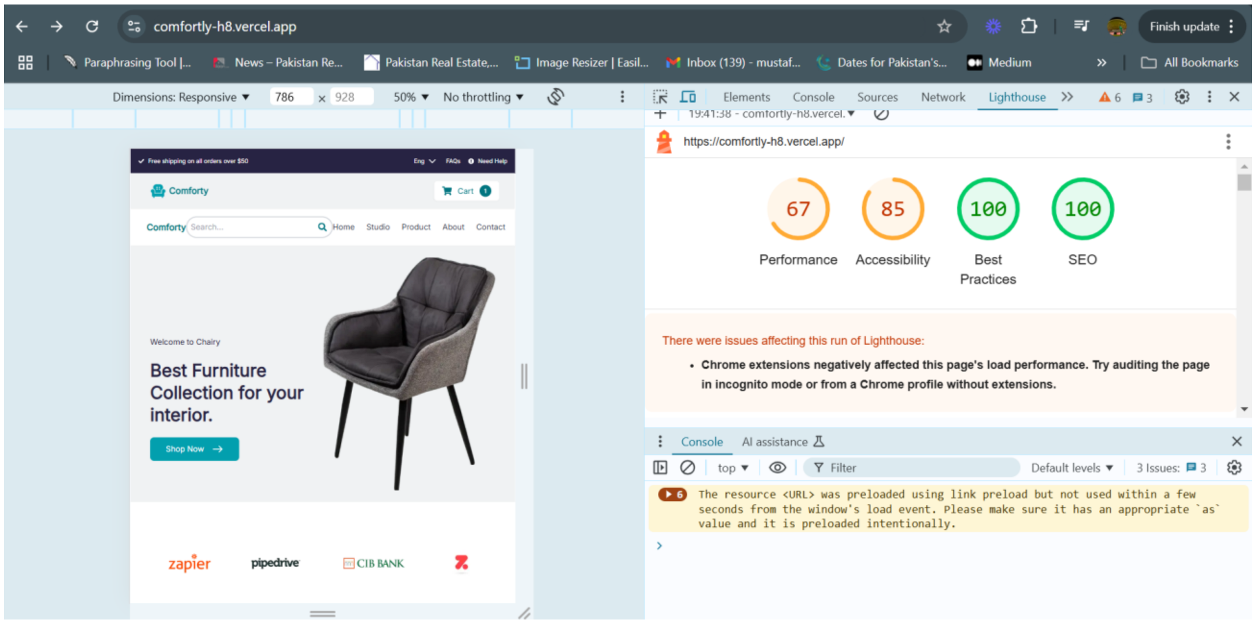
⚠ Minor Consideration:

- Continuously monitor structured data validation.

---

3. Action Plan

1. **Improve Performance**
   - Optimize and compress assets.
   - Use lazy loading and proper caching strategies.
   - Minimize render-blocking resources.
2. **Enhance Accessibility**
   - Ensure proper labeling of elements.
   - Improve contrast where necessary.
3. **Maintain Best Practices and SEO**
   - Continue security updates.
   - Monitor structured data and SEO trends.


comfortly-h8.vercel.app
Dimensions: Responsive 786 x 928 50% No throttling
Elements Console Sources Network Lighthouse
Comforty
Best Furniture Collection for your interior.
Shop Now

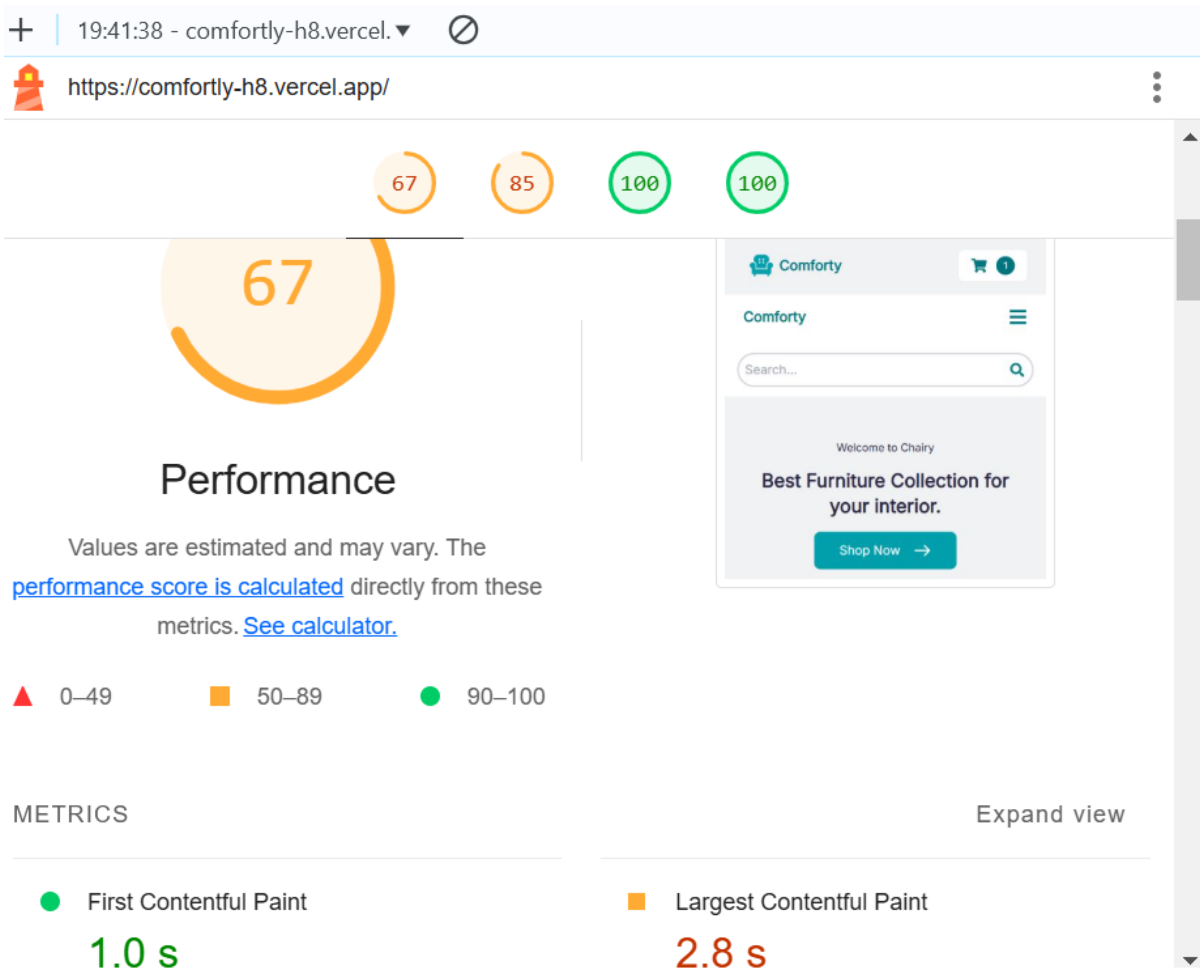
https://comfortly-h8.vercel.app/

Performance	Accessibility	Best Practices	SEO
67	85	100	100

There were issues affecting this run of Lighthouse:

- Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.

```
The resource <URL> was preloaded using link preload but not used within a few
seconds from the window's load event. Please make sure it has an appropriate `as`
value and it is preloaded intentionally.
```



19:41:38 - comfortly-h8.vercel.
https://comfortly-h8.vercel.app/
67
85
100
100
Comforty
Comforty
Search...
Welcome to Chairy
Best Furniture Collection for your interior.
Shop Now

67

Performance

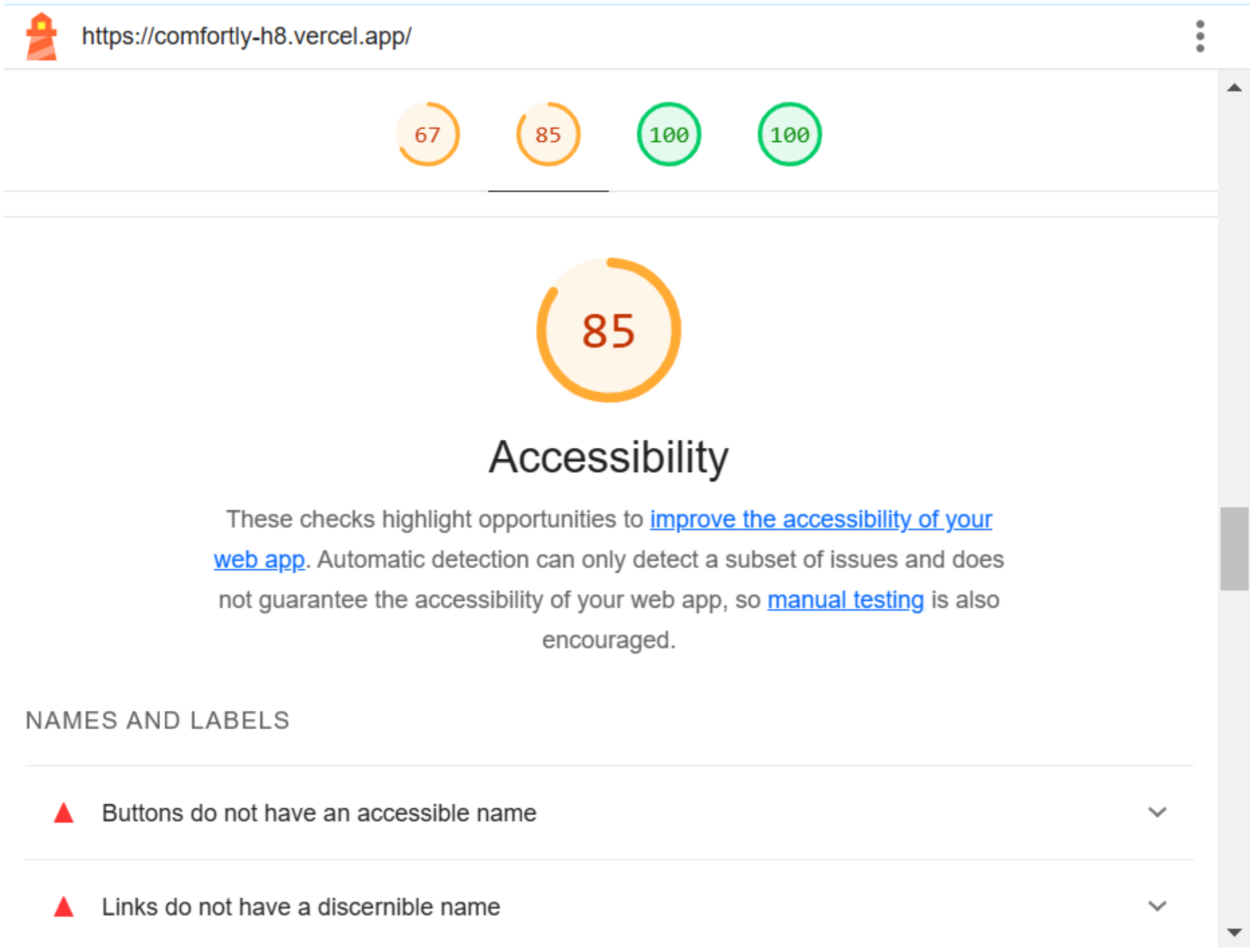
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint 1.0 s

■ Largest Contentful Paint 2.8 s



https://comfortly-h8.vercel.app/

67 85 100 100

85

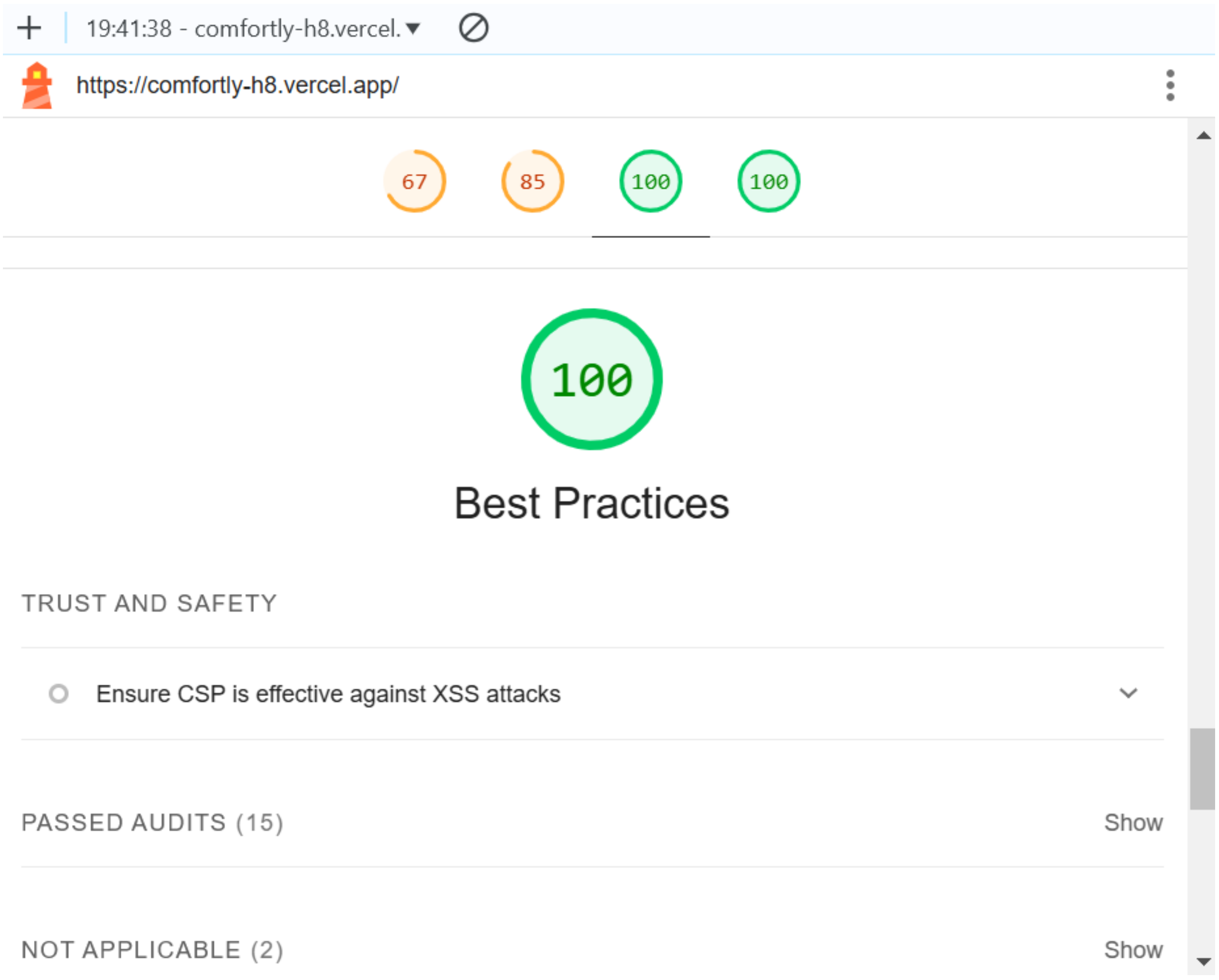
Accessibility

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

▲ Links do not have a discernible name



19:41:38 - comfortly-h8.vercel.

https://comfortly-h8.vercel.app/

67 85 100 100

100

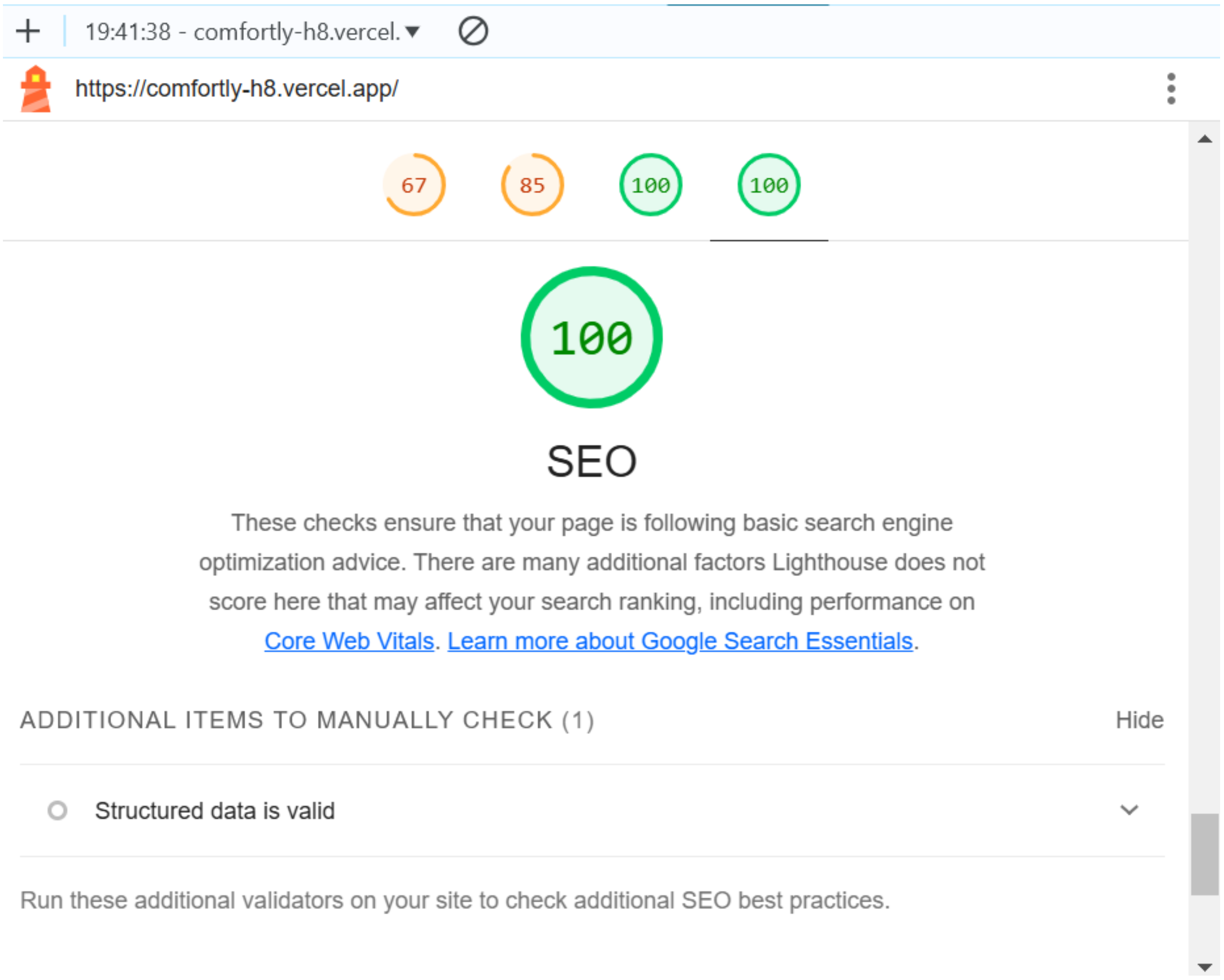
Best Practices

TRUST AND SAFETY

Ensure CSP is effective against XSS attacks

PASSED AUDITS (15) Show

NOT APPLICABLE (2) Show


19:41:38 - comfortly-h8.vercel.
https://comfortly-h8.vercel.app/
67
85
100
100

100

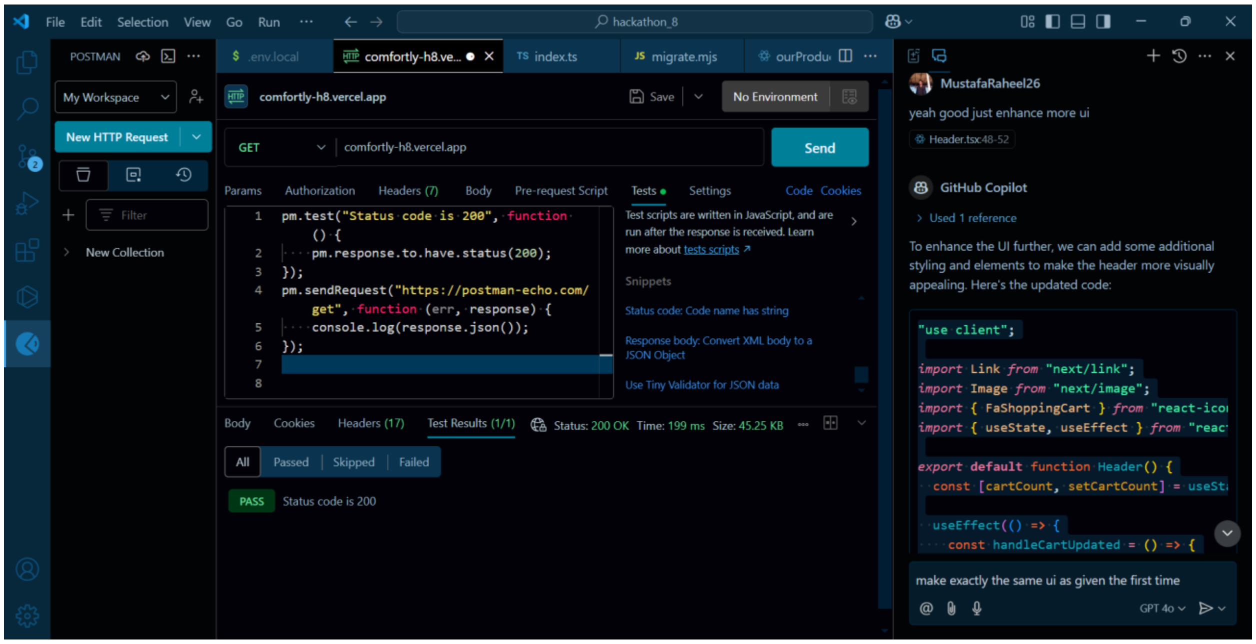
SEO

These checks ensure that your page is following basic search engine optimization advice. There are many additional factors Lighthouse does not score here that may affect your search ranking, including performance on Core Web Vitals. Learn more about Google Search Essentials.

ADDITIONAL ITEMS TO MANUALLY CHECK (1) Hide

Structured data is valid

Run these additional validators on your site to check additional SEO best practices.



comfortly-h8.vercel.app

GET comfortly-h8.vercel.app

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.sendRequest("https://postman-echo.com/get", function (err, response) {
    console.log(response.json());
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

Status code: Code name has string

Response body: Convert XML body to a JSON Object

Use Tiny Validator for JSON data

Status: 200 OK Time: 199 ms Size: 45.25 KB

PASS Status code is 200

MustafaRaheel26

yeah good just enhance more ui

Header.tsx:48-52

GitHub Copilot

Used 1 reference

To enhance the UI further, we can add some additional styling and elements to make the header more visually appealing. Here's the updated code:

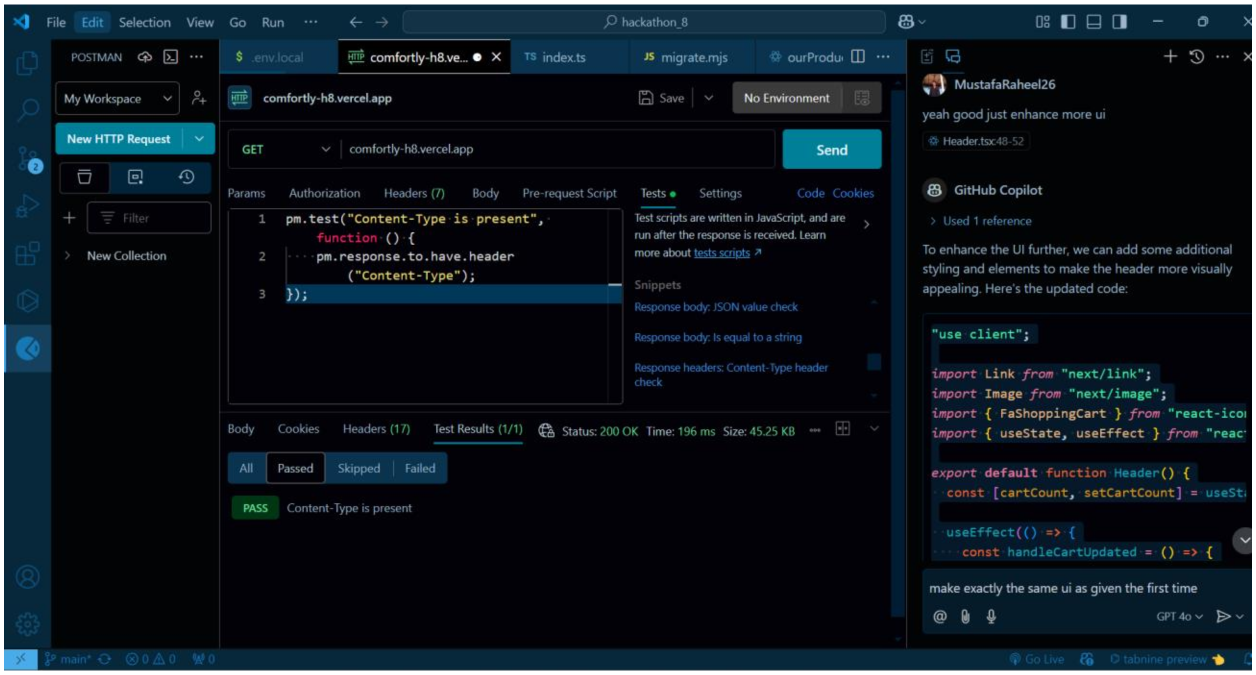
```
"use client";

import Link from "next/link";
import Image from "next/image";
import { FaShoppingCart } from "react-ico
import { useState, useEffect } from "reac

export default function Header() {
  const [cartCount, setCartCount] = useSt

  useEffect(() => {
    const handleCartUpdated = () => {
```

make exactly the same ui as given the first time



comfortly-h8.vercel.app

GET comfortly-h8.vercel.app

```
pm.test("Content-Type is present",
    function () {
    pm.response.to.have.header
        ("Content-Type");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Status: 200 OK Time: 196 ms Size: 45.25 KB

PASS Content-Type is present

MustafaRaheel26

yeah good just enhance more ui

Header.tsx:48-52

GitHub Copilot

Used 1 reference

To enhance the UI further, we can add some additional styling and elements to make the header more visually appealing. Here's the updated code:

```
"use client";

import Link from "next/link";
import Image from "next/image";
import { FaShoppingCart } from "react-icon
import { useState, useEffect } from "reac

export default function Header() {
  const [cartCount, setCartCount] = useSt

  useEffect(() => {
    const handleCartUpdated = () => {
```

make exactly the same ui as given the first time

2. *Performance Optimization Steps Taken:*

1. **Optimized API Requests**:
    a. Implemented lazy loading for product images and data to reduce initial load time.
    b. Consolidated multiple API calls into batch requests to minimize overhead.
2. **Caching Mechanisms**:
    a. Introduced client-side caching for frequently accessed data such as product details.
    b. Leveraged browser storage (localStorage) for storing user preferences and session data.
3. **Code Optimization**:
    a. Minimized JavaScript bundle size by removing unused dependencies and applying tree-shaking.
    b. Reduced CSS file size by adopting modular styles and purging unused classes.
4. **Load Testing**:

a. Conducted load testing to ensure the application performs well under concurrent user traffic.

---

3. *Security Measures Implemented:*

1. **Authentication and Authorization**:
   - Implemented JWT-based authentication to secure user sessions.
   - Restricted access to sensitive API endpoints based on user roles.
2. **Input Validation**:
   - Sanitized user inputs to prevent SQL injection and XSS attacks.
   - Utilized server-side validation for critical forms (e.g., login, registration).
3. **Secure Data Handling**:
   - Enforced HTTPS across all pages for secure communication.
   - Stored sensitive information, such as passwords, in hashed format using bcrypt.
4. **Vulnerability Scanning**:
   - Conducted regular scans using tools like OWASP ZAP to identify and mitigate vulnerabilities.

---

4. *Challenges Faced and Resolutions Applied:*

1. **Challenge**: Slow page loading due to large product images.
   - **Resolution**: Introduced image compression and lazy loading for non-critical assets.
2. **Challenge**: API downtime affecting functionality.
   - **Resolution**: Added fallback UI with meaningful error messages and retry logic for API calls.
3. **Challenge**: Ensuring cross-browser compatibility.

   - **Resolution**: Tested on multiple browsers and applied polyfills for unsupported features.
4. **Challenge**: Maintaining responsive design for mobile users.
   - **Resolution**: Utilized a mobile-first CSS framework and thoroughly tested on various screen sizes.