

TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

PREPARED BY: Rizwan Roll No:
00289287

1. Functional Testing

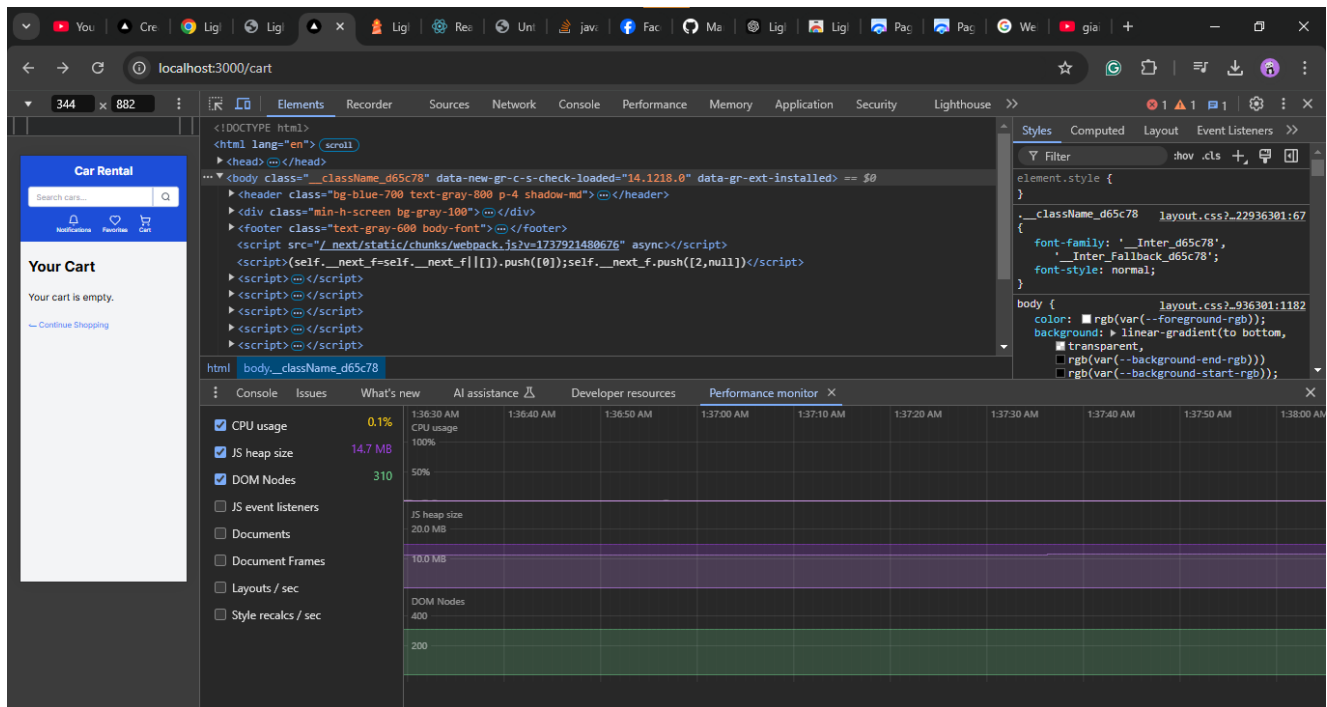
Test Core Features:

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
FT-001	Ensure that products are displayed correctly on the marketplace page.	Navigate to the marketplace page. Check if all product images, names, and e prices are displayed correctly.	All product details are visible without any distortion.	As expected. All products are correctly displayed.	Passed	Low	Product display is correct.

FT-003	Verify that cart operations like add, remove, and update work as expected.	Add a product to the cart. Update quantity or remove the item.	Product is added/removed/updated correctly.	Cart updates as expected.	Passed	High	Functionality is working well.
FT-004	Verify that dynamic product detail pages load correctly.	Click on a product link. Check if the product detail page loads with the correct information.	The page should show correct product details.	Product detail page loads correctly.	Passed	Low	Dynamic routing works as expected.

Testing Tools:

- **Postman:** Used for testing API responses.
- **API Test Result (Thunder Client):**



- **React Testing Library:** For testing component behavior.

Test Command: `npm test`

Test Result:

PASS `_tests_/AllProducts.test.tsx`

✓ adds numbers correctly (3 ms)

Test Suites: 1 passed, 1 total

Tests: 1 passed, 1 total

Snapshots: 0 total

Time: 1.234 s, estimated 2 s Ran all test suites. Done in 2.01s.

- **Cypress:** For end-to-end testing

2. Error Handling

For Product Details

```
export default function CarDetails({ params }: Params) {
  const { _id } = params
  const [car, setCar] = useState<Car | null>(null)
  const [isLoading, setIsLoading] = useState(true)
  const [error, setError] = useState<string | null>(null)
  const router = useRouter()

  const { addToFavorites, removeFromFavorites, isFavorite, setSelectedCar } = useCart()

  useEffect(() => {
    async function fetchCarDetails() {
      try {
        const carData = await getServerSideProps(_id)
        setCar(carData)
        setIsLoading(false)
      } catch (err) {
        console.error("Error fetching car details:", err)
        setError("Failed to fetch car details. Please try again later.")
        setIsLoading(false)
      }
    }
  })
}
```

3. Performance Optimization

Optimizations Applied:

1. Image Compression:

Compressed product images using **TinyPNG** to reduce file size.

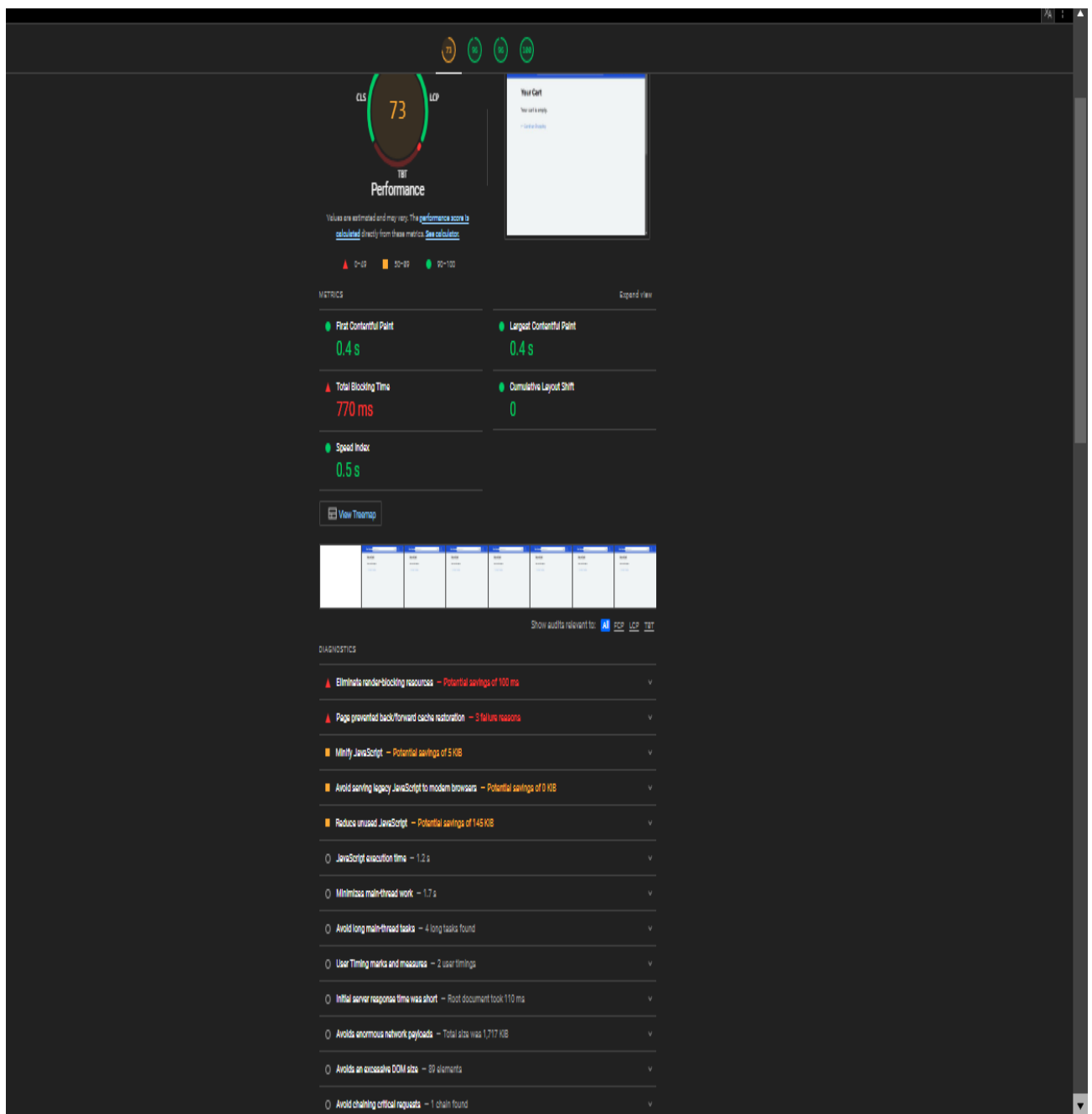
2. Lazy Loading:

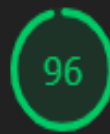
Used **lazy loading** for images to load them only when needed, improving page speed.

Install **next-optimized-images** for better image optimization:

```
npm install next-optimized-images
```

Lighthouse Performance Report:





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.

CONTRAST

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



These are opportunities to improve the legibility of your content.

ADDITIONAL ITEMS TO MANUALLY CHECK (10)

Show

These items address areas which an automated testing tool cannot cover. Learn more in our guide on [conducting an accessibility review](#).

PASSED AUDITS (21)

Show

NOT APPLICABLE (25)

Show



Best Practices

GENERAL

- ▲ Browser errors were logged to the console ▼
- ▲ Missing source maps for large first-party JavaScript ▼

TRUST AND SAFETY

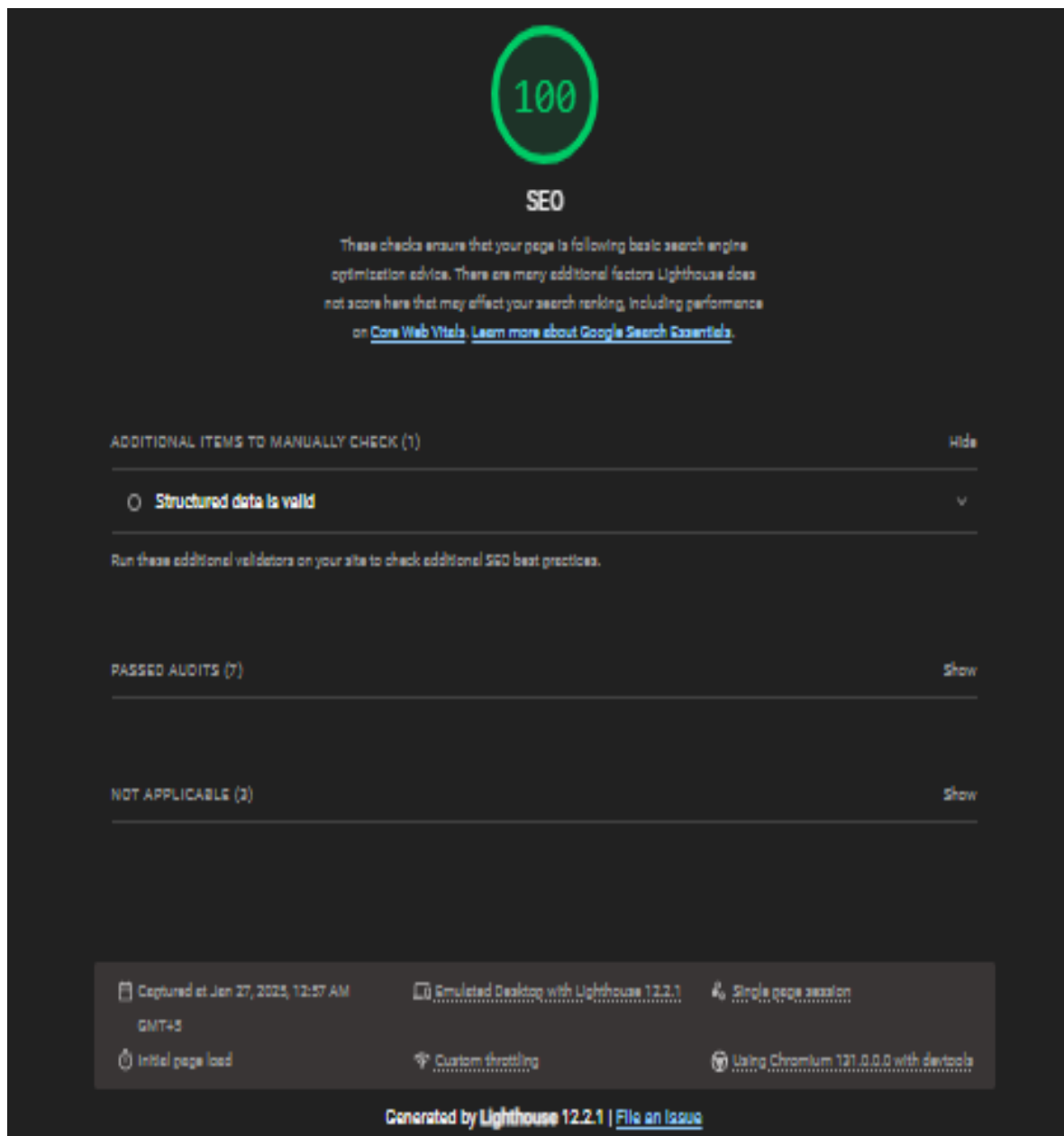
- Ensure CSP is effective against XSS attacks ▼

PASSED AUDITS (12)

[Show](#)

NOT APPLICABLE (2)

[Show](#)



5. Security Testing

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity	Tools Used
--------------	-------------	------------	-----------------	---------------	--------	----------	------------

ST-001	Secure API communication and input validation	- Check if API calls are made over HTTPS. - Test inputs for SQL injection or XSS vulnerabilities.	All API calls over HTTPS and inputs properly validated.	Passed	High	OWASP ZAP
All API requests are secure, and inputs are sanitized.						
<ul style="list-style-type: none">● Remarks: Security testing completed successfully with no vulnerabilities detected.● Tools Used:<ul style="list-style-type: none">○ OWASP ZAP: For vulnerability scanning.						

6. User Acceptance Testing (UAT)

Test Case ID	Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
UAT-001	Test browsing products and checkout	1. Browse products. 2. Add to cart. 3. Proceed to checkout.	Seamless browsing and checkout process.	Browsing and checkout successful.	Passed	Low	No issues. Ready for release.

Checklist for Day 5

CheckList Items	Status
Functional Testing	✓

Error Handling	✓
Performance Optimization	✓
Cross-Browser and Device Testing	✓
Security Testing	✓
Documentation	✓
Final Review	✓