DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

1. **Functional Testing**
    - Product listing: Products are displayed correctly.
    - Filters and search: Validate accurate results based on user inputs.
    - Cart operations: Add, update, and remove items from the cart.
    - Dynamic routing: Verify individual product detail pages load correctly.

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC001	Products are displayed correctly	Open Products Page > Validate Products	Products Display Successfully	Products Display Successfully	Low	-	No Issue Found
TC002	Validate accurate results based on user inputs	On searching it filters product accurately and smoothly.	Product Search successfully	Product Search successfully	low	-	Successfully searched
TC003	Add, update, and remove items from the cart.	Add to cart → signin → add to cart → notification → cart item length increase → click delete icon → notification for delete confirmation → remove → item length will be zero.	Product Add to cart and remove from cart successfully done.	Product Add to cart and remove from cart successfully done.	low	-	Gracefully handle adding and removing
TC004	Verify individual product detail pages load correctly.	Navigate to shop and on click details button → it will navigate to dynamic route (single-product) page and show the product details with slug url.	Navigate to dynamic route and show details of product successfully.	Display product details easily.	medium	-	Successfully display and no issue found.

Our Products

Modern Floor Lamp
170   ~~200~~
Stock : 15

Antique Floor Lamp
110
Available for Rent   Stock : 20

Luxury Bedroom Chair
171   ~~190~~
Available for Rent   Stock : 50

Modern Side Table
162   ~~180~~
Out of Stock

Modern TV Unit
160   ~~200~~
Out of Stock

King-Size Bed
90   ~~100~~
Available for Rent   Stock : 5

Wooden Desk
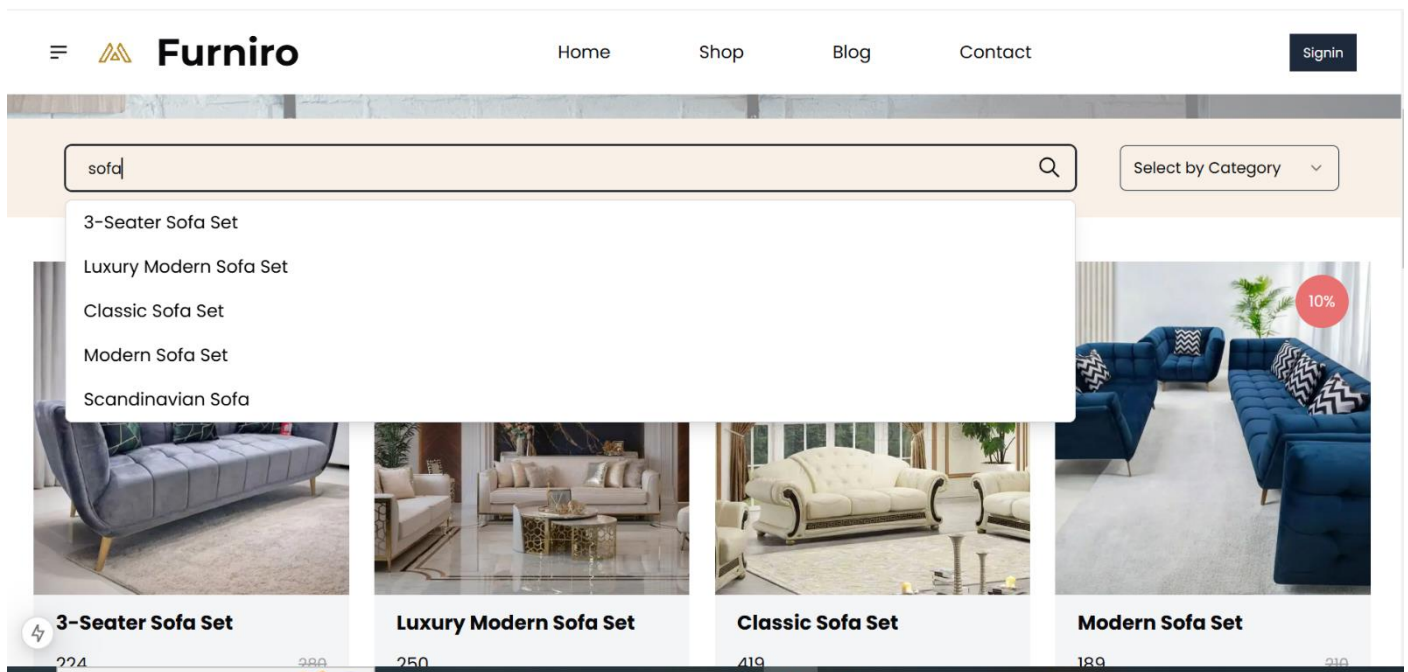340
Out of Stock

Glass Top Coffee Table
200
Available for Rent   Out of Stock
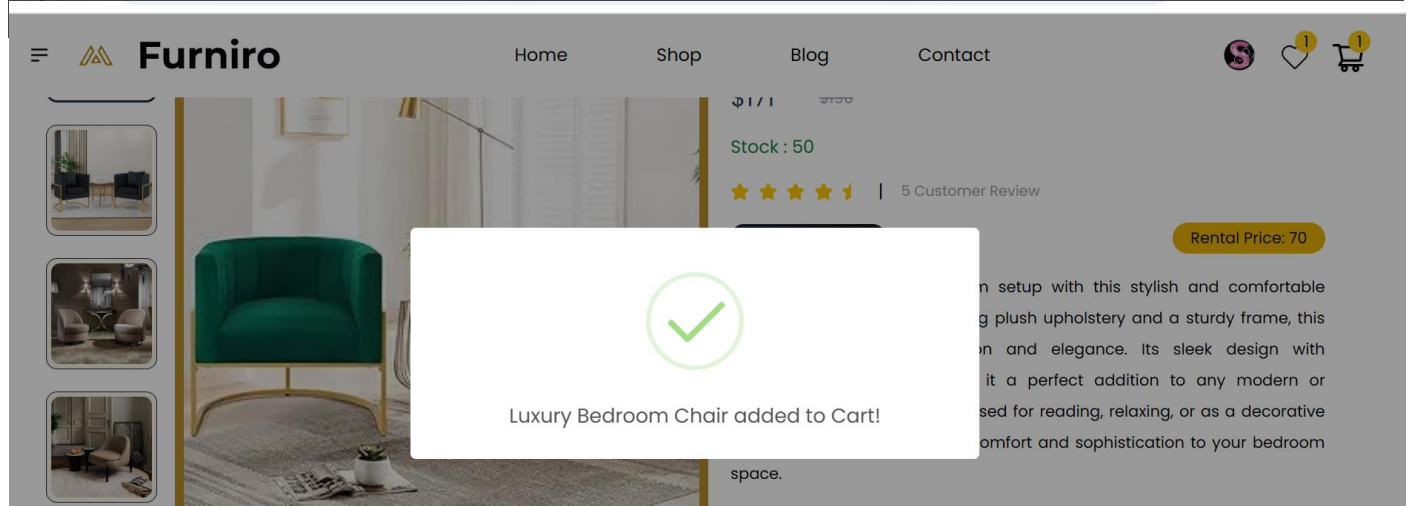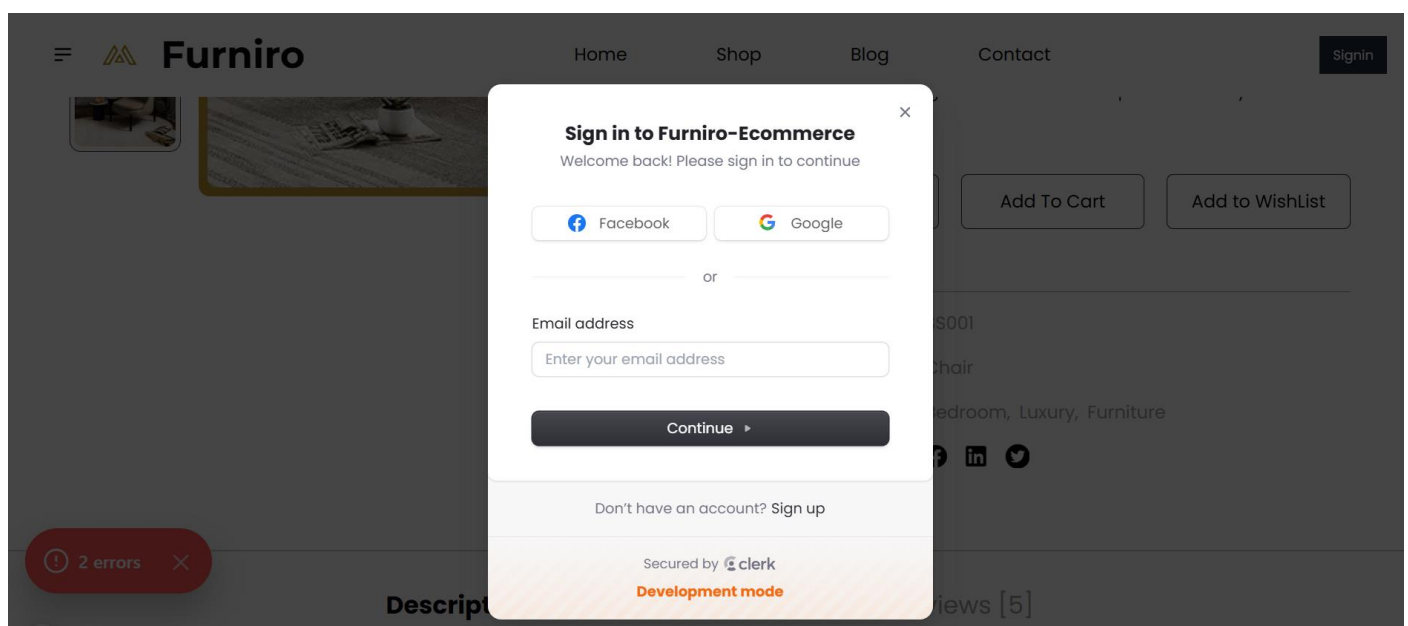
Show More

Filters and search: Validate accurate results based on user inputs.

Add, update, and remove items from the cart.

Go to product → product details page (dynamic route) → add to cart → sign in → then add to cart → product will add to cart successfully → add to cart quantity increase according to product length. On click delete icon the product will remove from cart.

Furniro

Home   Shop   Blog   Contact

Cart

Product	Title	Price	Quantity	Subtotal	
	Luxury Bedroom Chair	190	1	190	🗑

Cart Totals

Subtotal	Rs. 190
Total	**Rs. 190**

Check Out

Furniro

Home   Shop   Blog   Contact

!

Are You Sure?

You want to remove this item

Yes, Remove it.   Cancel

Product	Title	Price		

| | Luxury Bedroom Chair | 190 | 1 | 190 | 🗑 |

Cart Totals

Subtotal	Rs. 190
Total	**Rs. 190**

Check Out

Furniro

Home   Shop   Blog   Contact

Cart

Product	Title	Price	Quantity	Subtotal

Your Cart is Empty

Cart Totals

Subtotal	Rs. 0
Total	**Rs. 0**

Check Out

Testing Tools:

- **Postman:** For API response testing.

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC005	Testing API in Postman	Paste API URI in Postman (GET)	API Fetch in JSON Format	API Fetched successfully	Low	-	No Issue Found

3. Performance Testing

Performance Metrics

The following metrics are generated using Lighthouse Performance data.

Metric details **OFF**

First Contentful Paint ?	Good - Nothing to do here **654ms**	Time to Interactive ?	Good - Nothing to do here **1.0s**
Speed Index ?	Good - Nothing to do here **818ms**	Total Blocking Time ?	Good - Nothing to do here **147ms**
Largest Contentful Paint ?	Good - Nothing to do here **1.0s**	Cumulative Layout Shift ?	Good - Nothing to do here **0**

Browser Timings

These timings are milestones reported by the browser.

Redirect Duration ?	399ms	Connection Duration ?	2ms	Backend Duration ?	119ms
Time to First Byte (TTFB) ?	520ms	DOM Interactive Time ?	561ms	DOM Content Loaded Time ?	562ms
First Paint ?	654ms	Onload Time ?	1.1s	Fully Loaded Time ?	1.5s

Step 2: Error Handling 1. Add Error Messages:

≡ ⚠ **Furniro**   Home   Shop   Blog   Contact   Signin

Oops! 😔

We're really sorry, but we couldn't find the product you were looking for.

Please try refreshing the page. We appreciate your patience! 🌸

Back to Shop

```
1  useEffect(() => {                                    36       } finally {
2    const fetchSingleProductData = async () => {       37         setLoading(false);
3      try {                                             38       }
4        if (slug) {                                     39     };
5          const groqQuery = ``;                         40     fetchSingleProductData();
6          const fetchedData = await client.fetch(groq`${groqQuery}`);   41   }, [slug]);
7          const findProduct = fetchedData.find(         42
8            (singleProductData: ProductType) =>         43   if (loading) {
9              singleProductData.slug.current === slug   44     return <LoadingPage />;
10         );                                            45   }
11         if (findProduct) {                            46
12           setSingleProduct([findProduct]);            47   if (error) {
13         } else {                                      48     return <Errpage error={error} />;
14           setSingleProduct([]);                       49   }
15         }                                             50
16       }                                               51   return (
17     } catch (error) {                                 52     <>
18       console.log("Error fetching product data:", error);   53       {singleProduct.map((singleFurniture: ProductType, index: number) => {
19       setError(                                       54         const discountPrice =
20         "We're really sorry, but we couldn't find the product you were looking for."   55           (singleFurniture.price * singleFurniture.dicountPercentage) / 100;
21       );                                              56         const discountedPrice = singleFurniture.price - discountPrice;
                                                         57
```
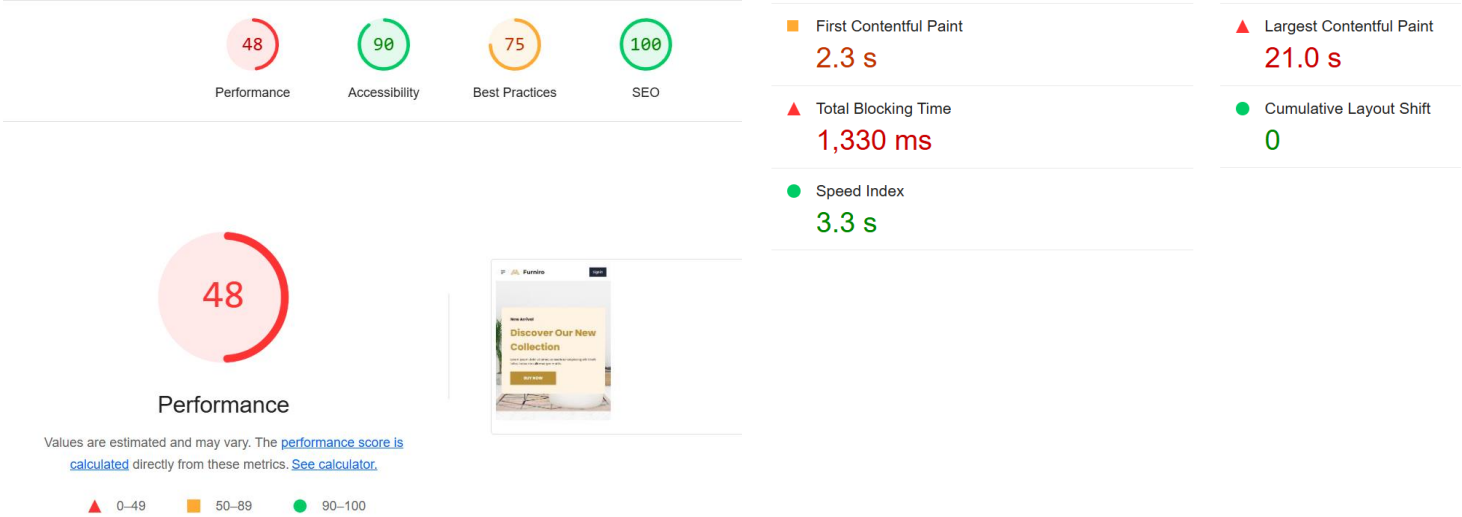
Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC006	Test API Error Handling	Remove Groq Query → Refresh Page	Error Message in the form of UI on Screen	Error Message in the form of UI on Screen	Low	-	Handle Gracefully

Step 3: Performance Optimization

Lighthouse performance Testing:

METRICS

First Contentful Paint
2.3 s

Largest Contentful Paint
21.0 s

Total Blocking Time
1,330 ms

Cumulative Layout Shift
0

Speed Index
3.3 s

48

Performance

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

▲ 0–49   ■ 50–89   ● 90–100

DIAGNOSTICS

▲ Avoid multiple page redirects — Potential savings of 1,840 ms   ⌄

▲ Serve images in next-gen formats — Potential savings of 715 KiB   ⌄

▲ Minify JavaScript — Potential savings of 539 KiB   ⌄

▲ Reduce unused JavaScript — Potential savings of 1,336 KiB   ⌄

▲ Efficiently encode images — Potential savings of 13 KiB   ⌄

■ Properly size images — Potential savings of 123 KiB   ⌄

■ Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB   ⌄

○ Initial server response time was short — Root document took 430 ms   ⌄

DIAGNOSTICS

▲ Largest Contentful Paint element — 21,020 ms   ⌄

▲ Reduce unused JavaScript — Potential savings of 1,336 KiB   ⌄

▲ Serve images in next-gen formats — Potential savings of 715 KiB   ⌄

▲ Minify JavaScript — Potential savings of 539 KiB   ⌄

▲ Avoid multiple page redirects — Potential savings of 1,840 ms   ⌄

▲ Efficiently encode images — Potential savings of 13 KiB   ⌄

■ Preload Largest Contentful Paint image — Potential savings of -490 ms   ⌄

■ Properly size images — Potential savings of 123 KiB   ⌄

■ Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB   ⌄

DIAGNOSTICS

▲ Minimize main-thread work — 4.4 s   ⌄

▲ Reduce JavaScript execution time — 2.7 s   ⌄

○ Avoid long main-thread tasks — 19 long tasks found   ⌄

○ Minimize third-party usage — Third-party code blocked the main thread for 140 ms   ⌄

○ Avoids an excessive DOM size — 461 elements   ⌄

DIAGNOSTICS

○ Avoid large layout shifts — 1 layout shift found   ⌄

More information about the performance of your application. These numbers don't directly affect the Performance score.

| Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC008	Light House Performance tesing	DevTools → inspect → lighthouse → Select the appropriate categories (Performance, Accessibility, Best Practices, SEO) →Generate Report	Performance : 80 Accessibility : 100 Best Practice : 25 SEO : 50	Performance : 47 Accessibility : 90 Best Practice : 75 SEO : 100	Medium		Optimized

3. Test Load Times:
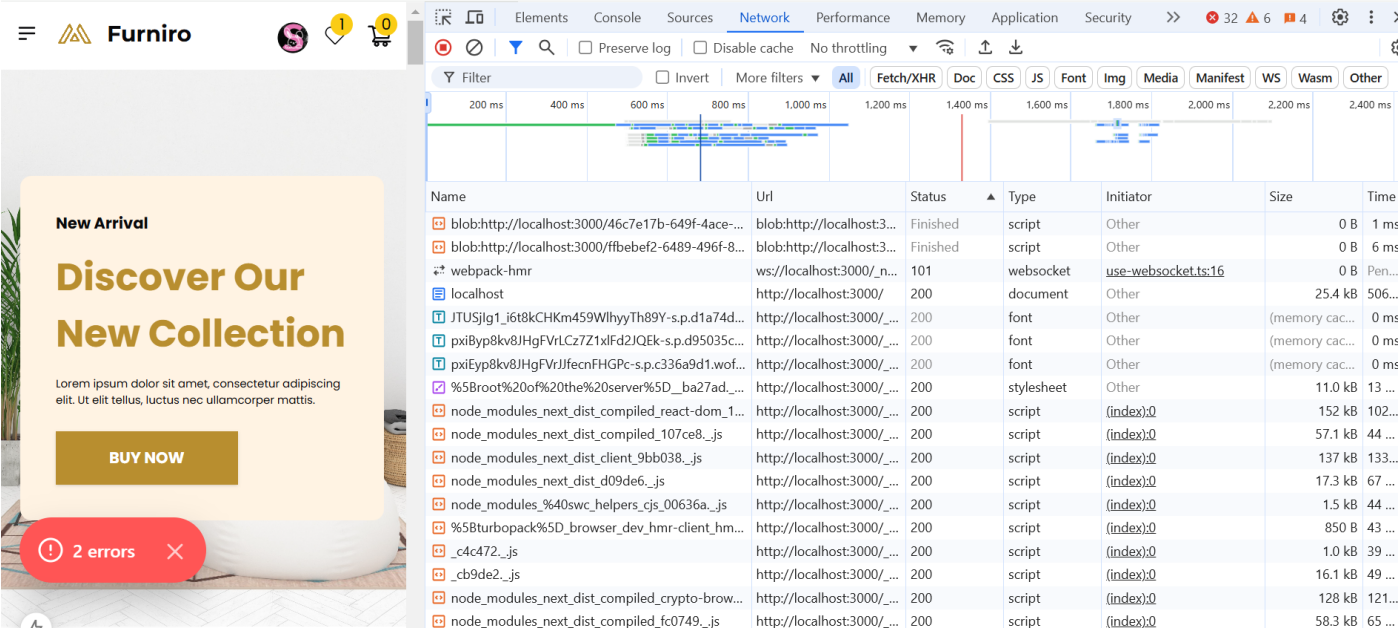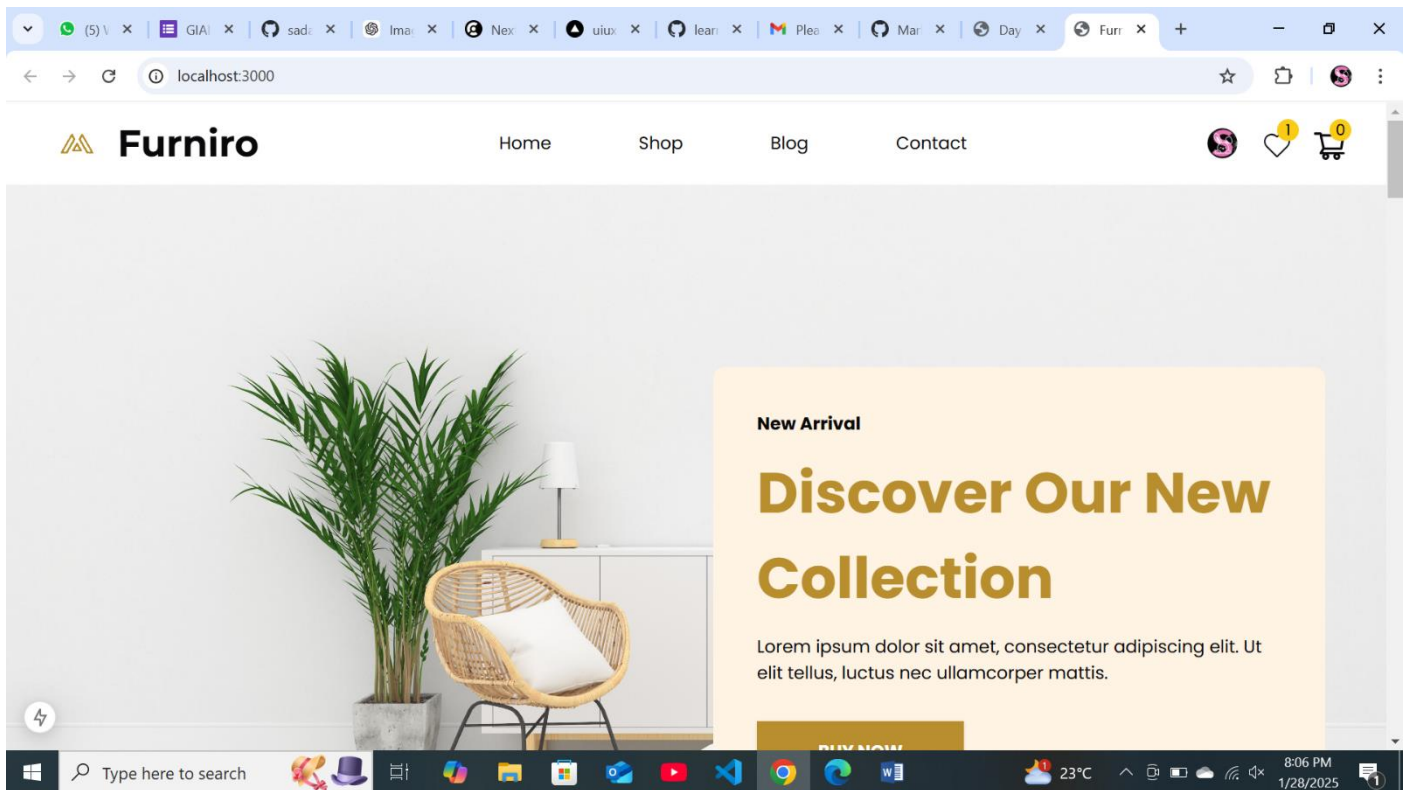

Developer resources:
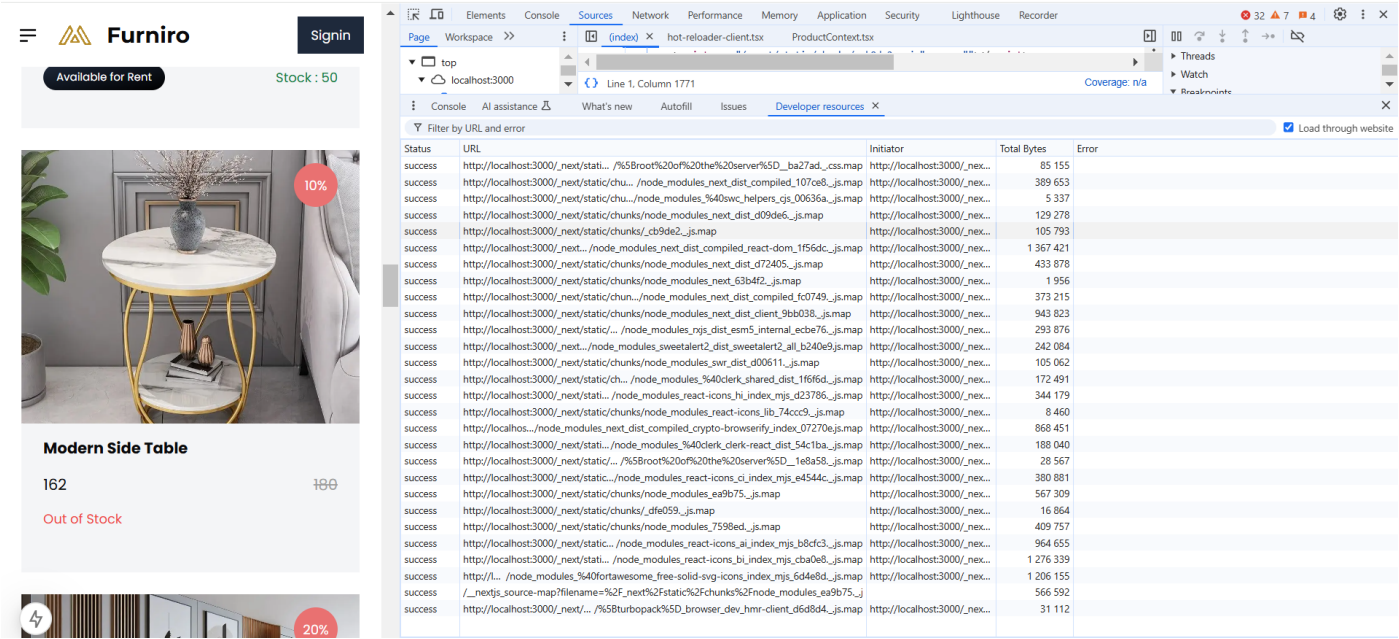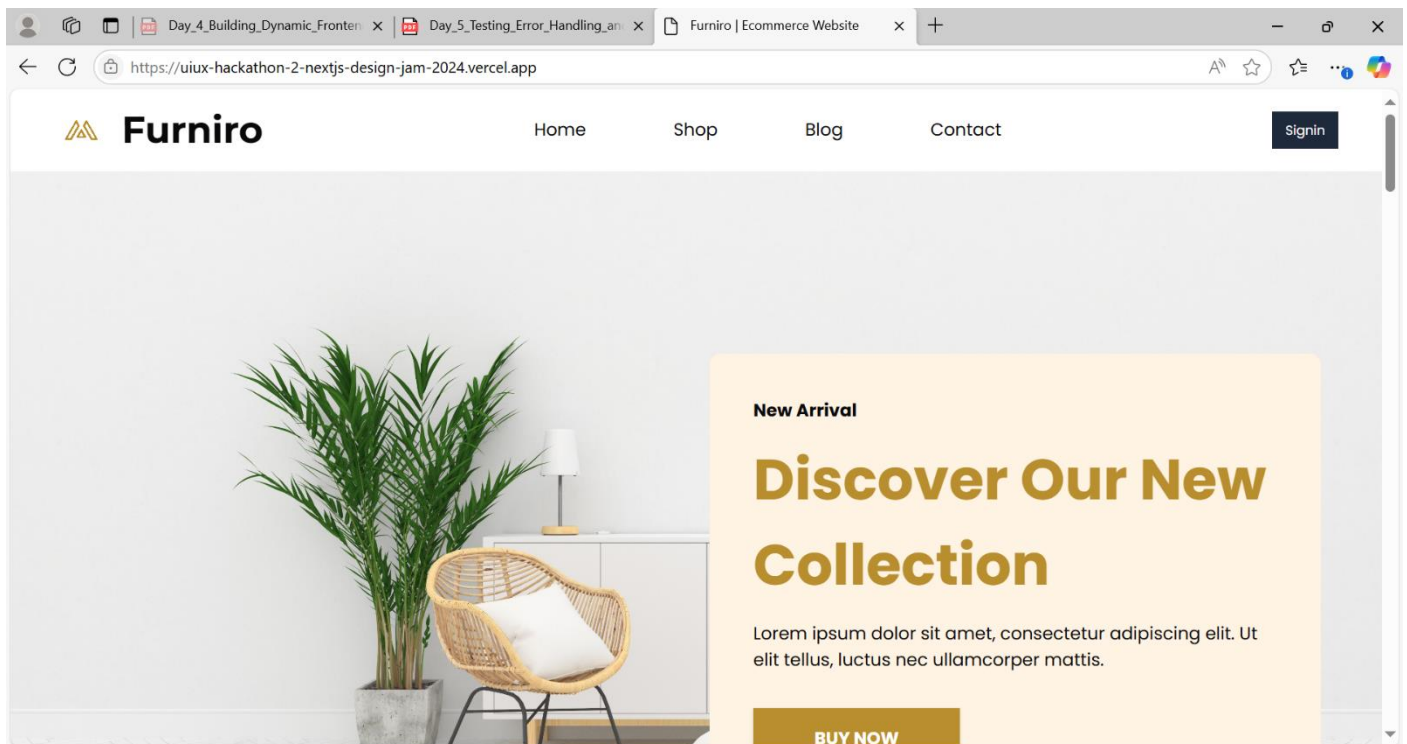

Performance Monitor:
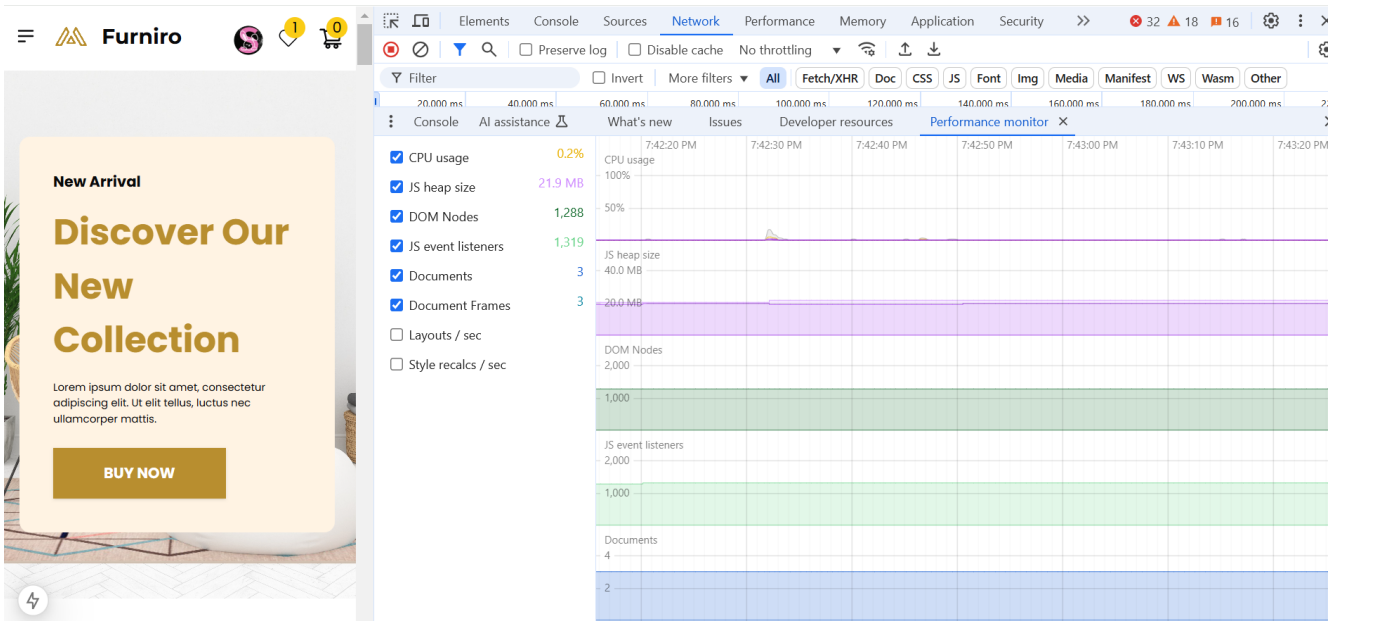
Step 4: Cross-Browser and Device Testing

Verify consistent rendering and functionality. --- Chrome
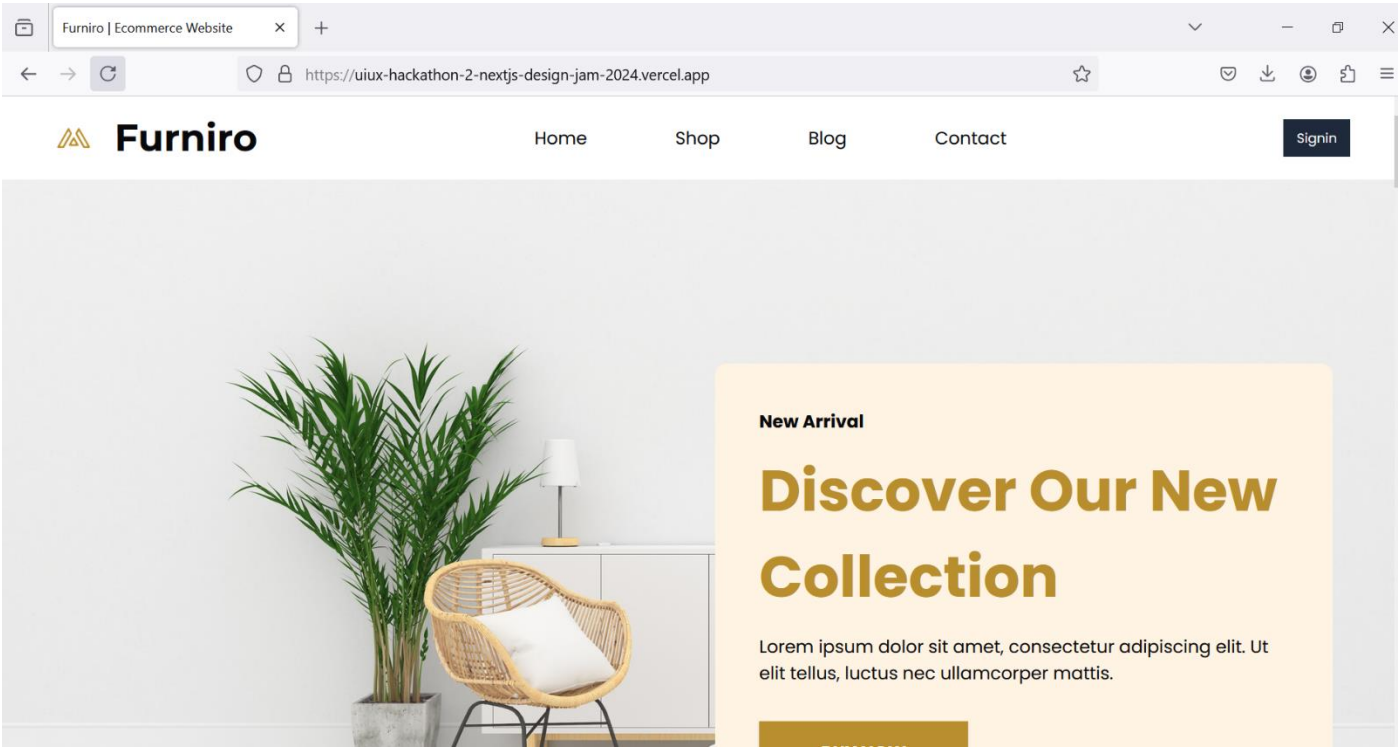
Test on Chrome, Firefox, Safari, and Edge.

Verify consistent rendering and functionality. --- Edge

Verify consistent rendering and functionality. --- FireFox Mozilla


Browser Testing Report:

- **Chrome**: All elements render correctly. No issues with layout, fonts, or functionality. Fast performance, no errors.
- **Firefox**: Similar to Chrome but slightly slower performance. Fonts render a bit differently, but no major issues.
- **Safari**: Minor lag in CSS transitions, but no rendering issues. Functions well overall.
- **Edge**: Rendering and functionality consistent with Chrome. Performance similar to Chrome with no issues.

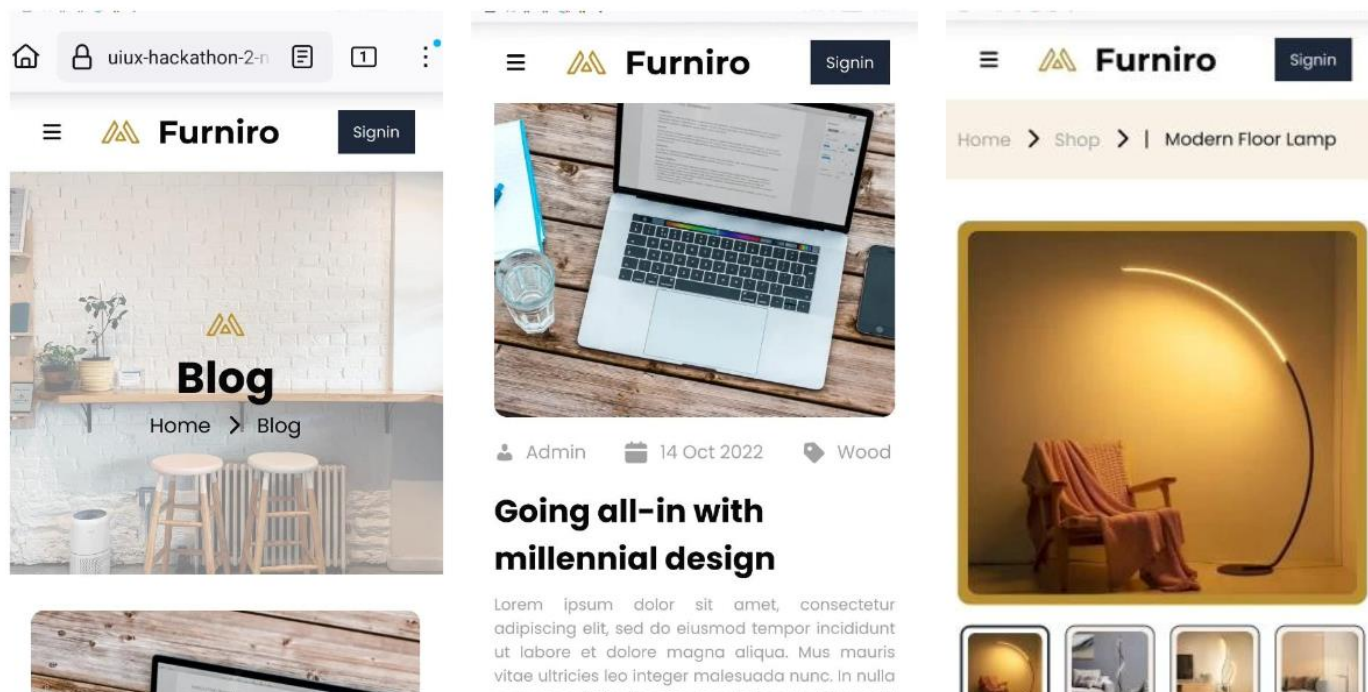| Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC009	• Browser Compatibility Testing	• Open the website in Chrome, Firefox, and Edge. • Verify layout, fonts, and functionality consistency across browsers. • Test responsiveness on different screen sizes. • Check for performance differences and errors in the browser console.	• Consistent rendering and functionality across all browsers. • No layout, font, or interactive element issues. • Fast performance with no console errors or significant delays.	• **Chrome**: All elements render correctly. Fast performance, no errors. • **Firefox**: Slightly slower performance; minor font rendering differences, but no major issues. • **Safari**: Minor lag in CSS transitions, no rendering or functional issues. • ⬜ **Edge**: Rendering and functionality consistent with Chrome. No issues.	Low	-	consistent and acceptable.

Device Testing:

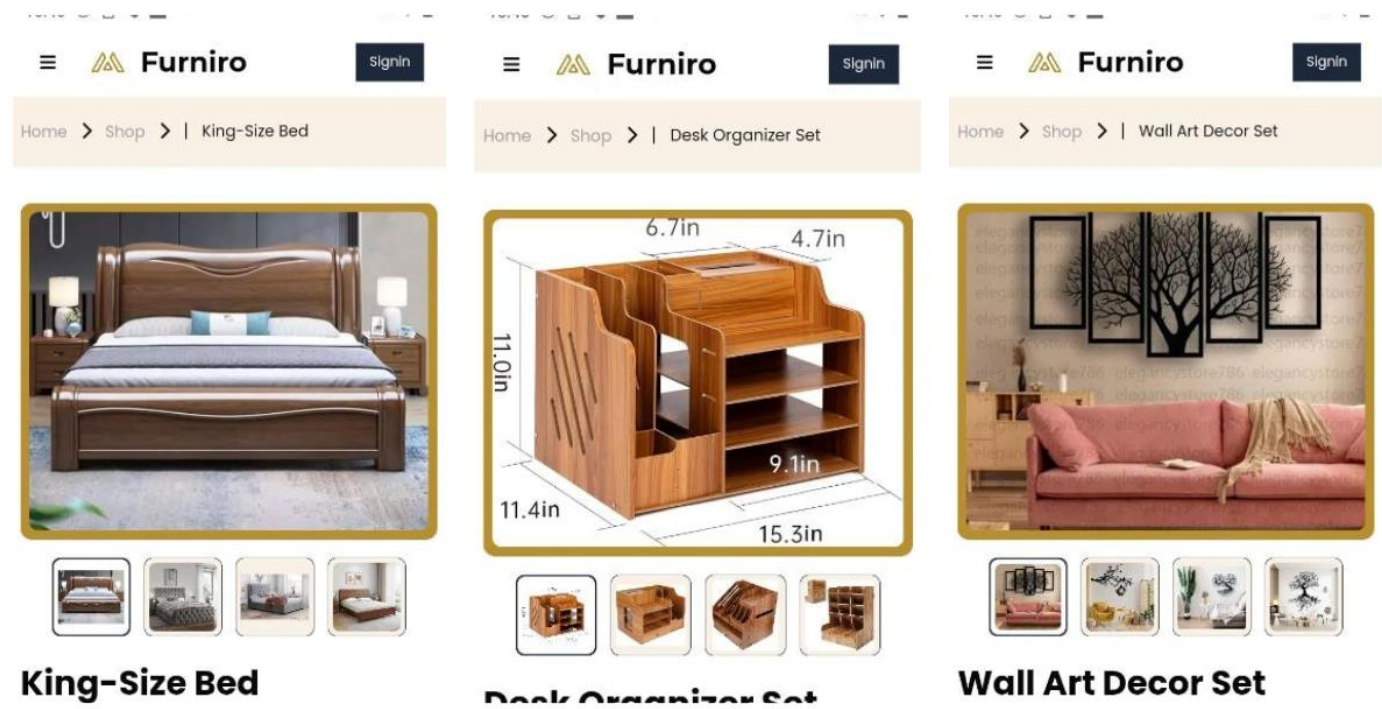Use responsive design tools like BrowserStack to simulate diAerent devices.

Google Mobile Responsive:
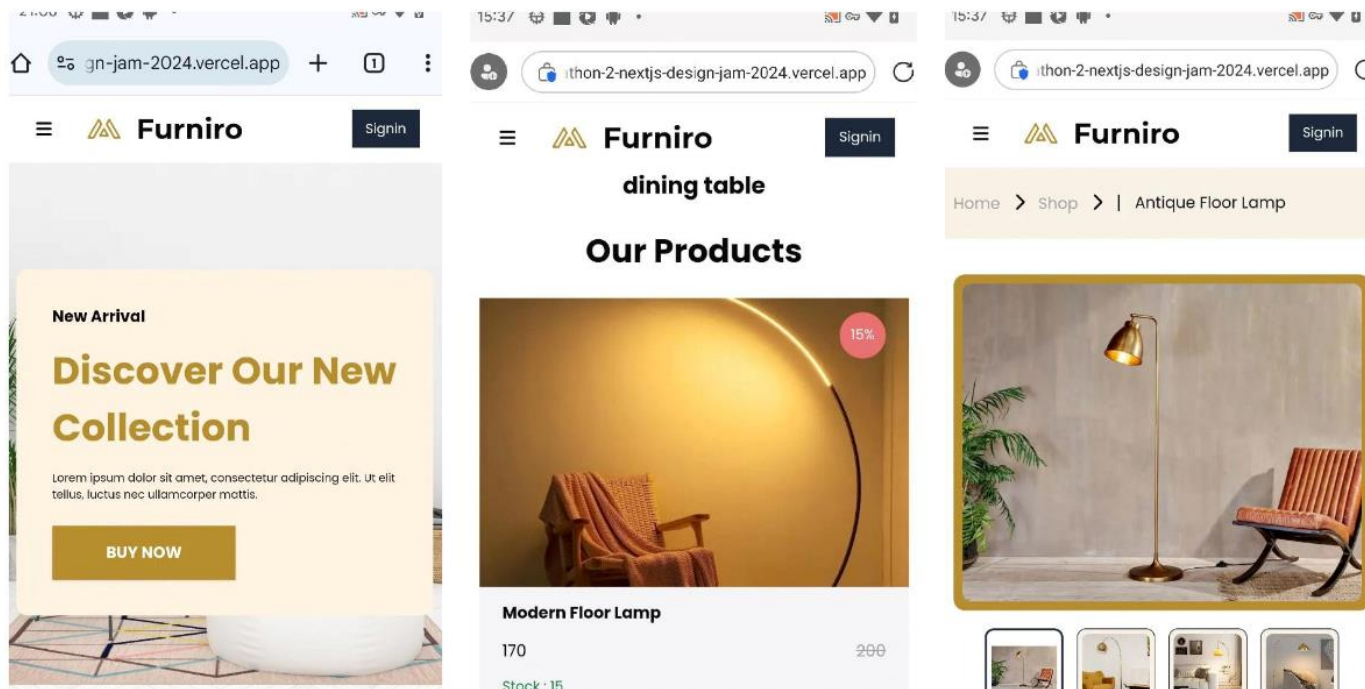
Huawei Mobile Responsive:

Moto Mobile Responsive:

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Severity Level	Assigned To	Remarks
TC009	Device Testing through BrowserStack	The layout, fonts, and interactive elements adapt correctly to all device sizes	Responsive across all devices with no layout or functionality issues	Responsive across all devices with no layout or functionality issues	Low	-	No functionality Issue Found.

Manually test on at least one physical mobile device.

Responsive with layouts and fonts adapt to my mobile screen size.

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