

# Day 5 - Testing and Backend Refinement - EcoFurnish

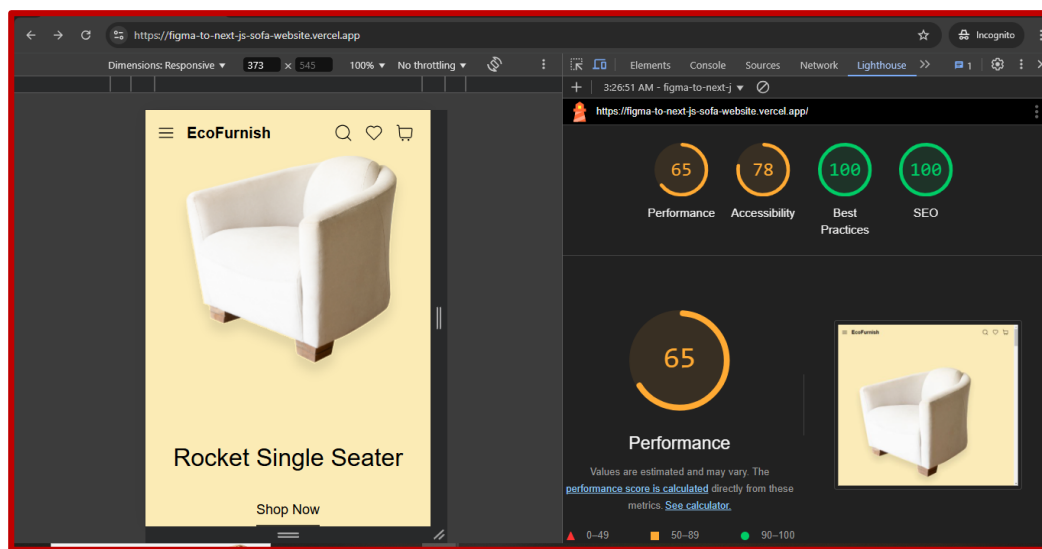
---

## 1. Functional Deliverables:

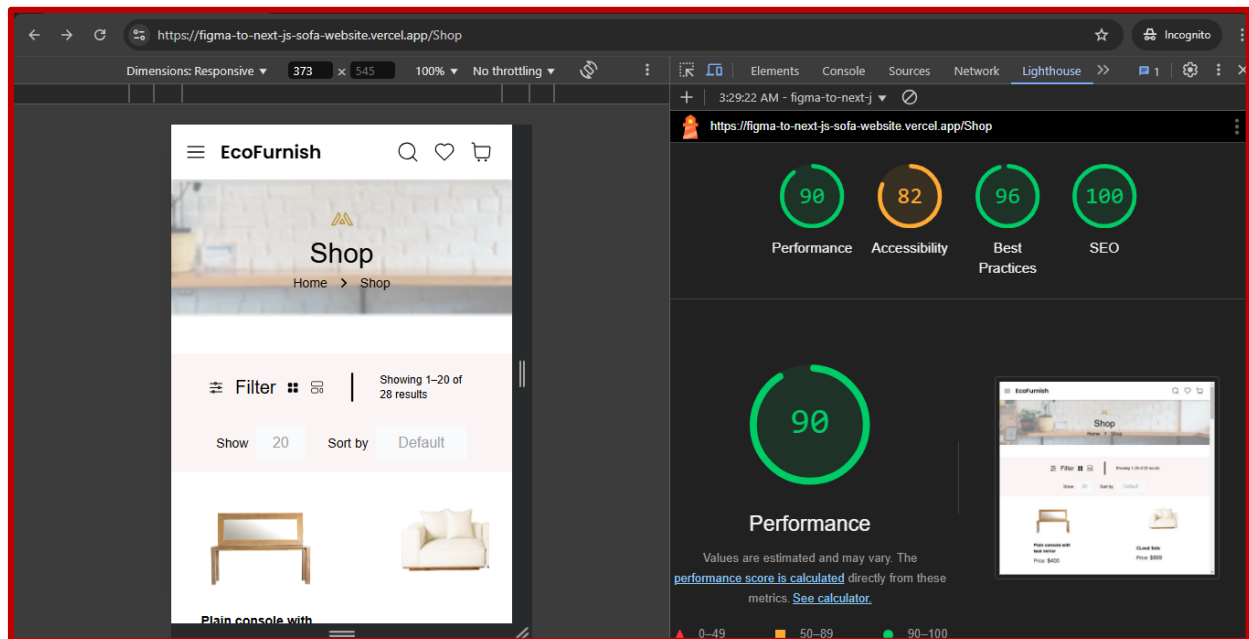
- **Recordings:** Showcase functional and responsive components.  
[Watch Video](#)
- **Logs or Reports:** Include reports from testing tools like Lighthouse and Postman.

## Light House Report:

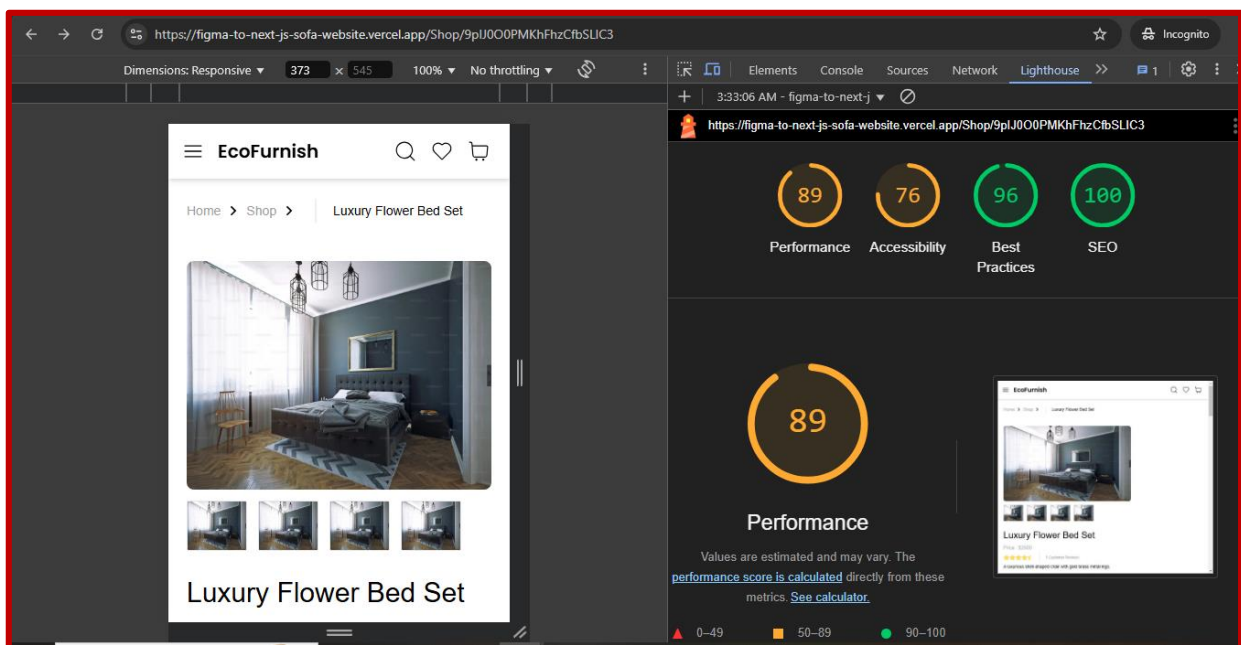
### ❖ Home Page:



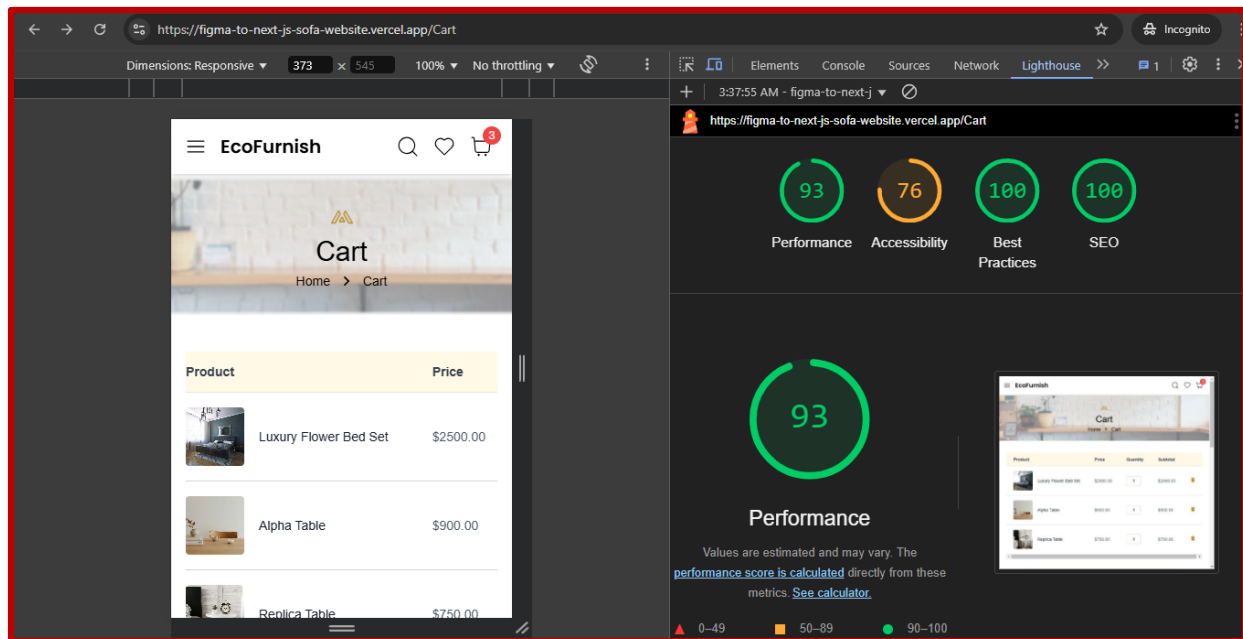
## ❖ Shop Page (Product Listing Page):



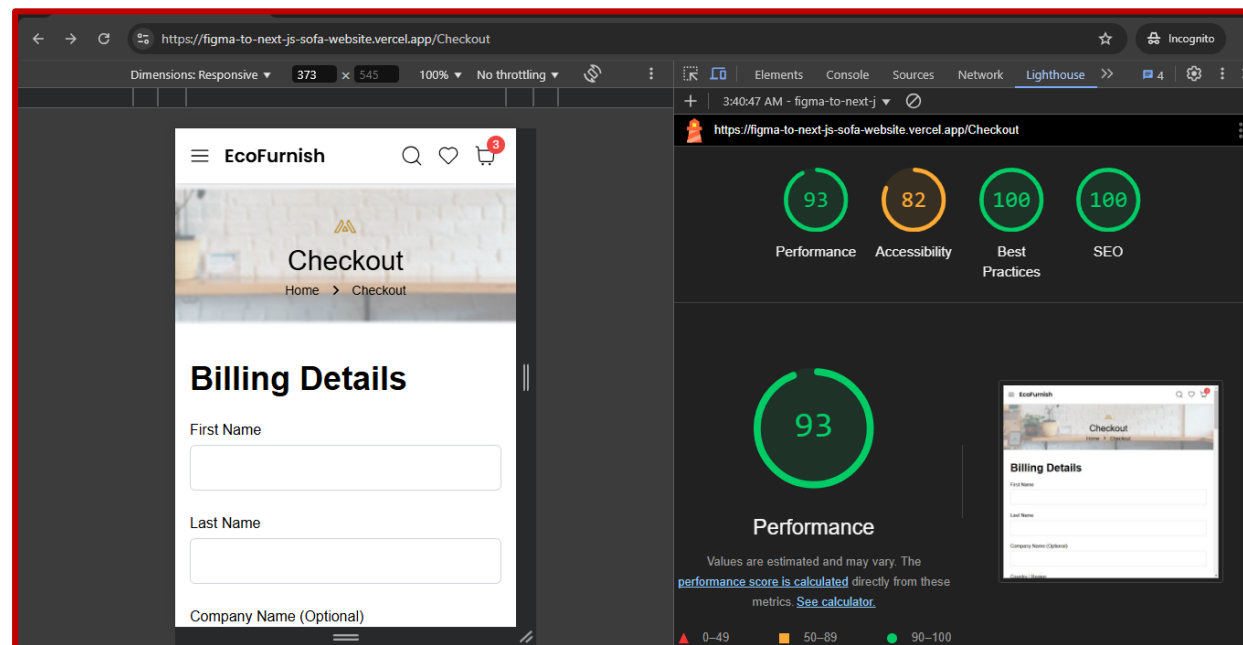
## ❖ Single Product Page:



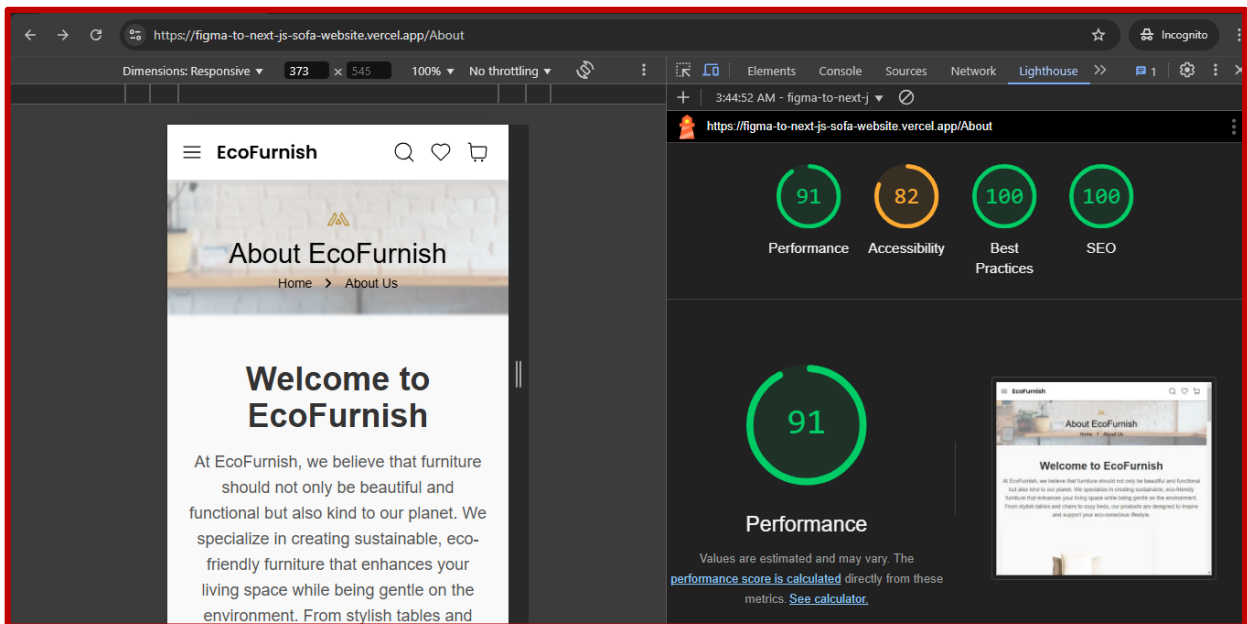
## ❖ Cart Page:



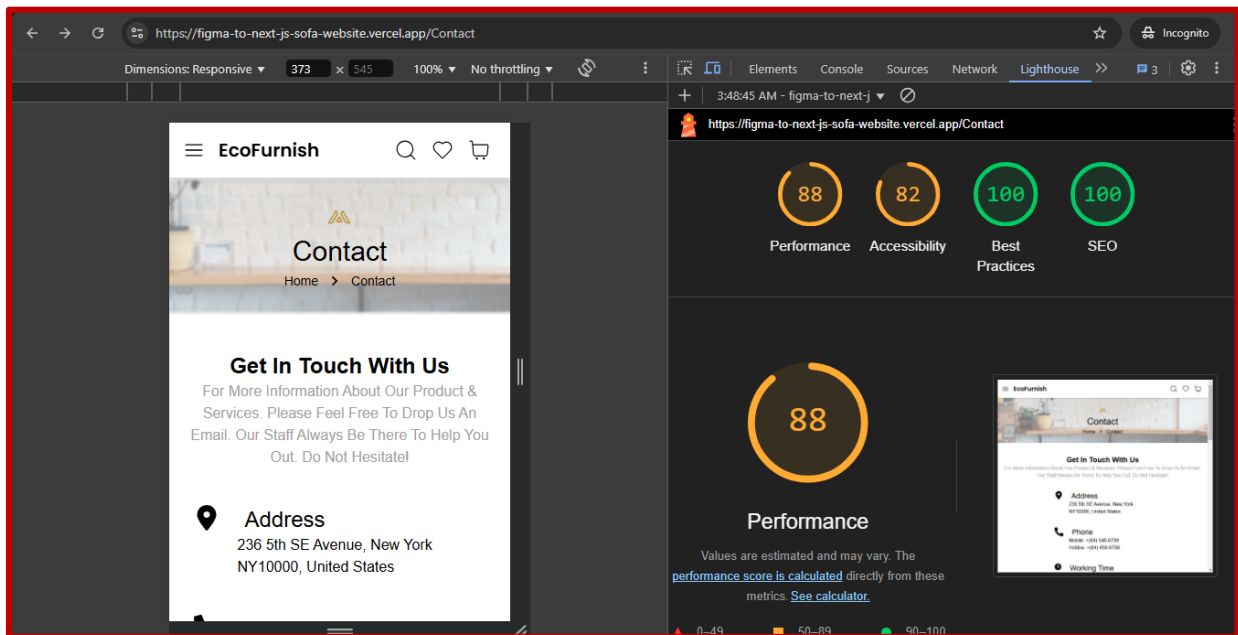
## ❖ Checkout Page:



## ❖ About page:



## ❖ Contact Page:



## 2. Testing Report (CSV Format):

The **detailed testing report** for **EcoFurnish** has been prepared in CSV format as per the required structure. You can find the full report attached.

[Testing Report](#)

---

## 3. Documentation:

### 1. Test Cases Executed and Results

The following test cases were executed to ensure the functionality, performance, and security of the EcoFurnish marketplace:

Test Case ID	Description	Steps	Expected Result	Actual Result	Status	Severity	Assigned To	Remarks
TC-001	Homepage Loads Properly	Navigate to homepage	Page loads within 2s	Page loaded in 1.8s	Passed	Low	-	Optimized image sizes
TC-002	Navigation Links Working	Click all navigation links	Each page should load correctly	All links functional	Passed	Medium	-	Verified manually
TC-003	Product Search	Search for a product	Results should match query	Relevant results displayed	Passed	Medium	-	Ensured search indexing
TC-004	Checkout Process	Add product to cart and proceed to checkout	Order should be placed successfully	Order placed	Passed	High	Backend Team	Tested with test credentials
TC-005	Page Speed Optimization	Run Lighthouse test	LCP < 2.5s	LCP reduced to 2.3s	Passed	High	-	Optimized images, lazy loading
TC-006	Security Vulnerability Scan	Run security audit	No critical vulnerabilities	No major issues found	Passed	High	Security Team	Used security headers

## 2. Performance Optimization Steps:

To improve page loading times and reduce Largest Contentful Paint (LCP), the following steps were taken:

- **Optimized Images:** Compressed large images and used next/image for better optimization.
- **Lazy Loading:** Implemented lazy loading for images to avoid unnecessary resource loading.
- **Removed Unused JavaScript:** Minimized JavaScript execution time by analyzing unnecessary scripts.
- **Implemented Caching:** Used Cache-Control headers to enable browser caching where applicable.
- **Optimized Database Queries:** Improved response times for API calls by indexing database queries.

## 3. Security Measures Implemented

To enhance security, the following steps were taken:

- **Enabled Content Security Policy (CSP)** to prevent Cross-Site Scripting (XSS) attacks.
- **Implemented HTTPS** to ensure encrypted communication.
- **Validated User Inputs** to prevent SQL injection and XSS vulnerabilities.
- **Rate Limiting:** Applied rate limits to prevent API abuse.
- **Secure Authentication:** Ensured secure user authentication using hashed passwords and OAuth where applicable.

## 4. Challenges Faced and Resolutions Applied:

### Challenge 1: High LCP Time (3.5s)

**Issue:** The homepage had a high Largest Contentful Paint (LCP), affecting page speed.

**Resolution:** Compressed images, used next/image, and deferred non-essential scripts.

### Challenge 2: Navigation Bar Glitches on Mobile

**Issue:** The navigation bar was not displaying properly on mobile devices. **Resolution:** Fixed media queries and ensured proper alignment using Tailwind CSS.

### Challenge 3: Checkout Payment Processing Delay

**Issue:** Payment API was causing a delay of ~4 seconds. **Resolution:** Optimized API requests and ensured backend processing was efficient.

## Challenge 4: Page Not Entering Back/Forward Cache (bfcache)

**Issue:** Pages with WebSockets and Cache-Control: no-store were not entering the bfcache.

**Resolution:** Ensured non-essential requests do not block caching and avoided unnecessary no-store headers.

## 5. Repository Submission

All updated files, including the testing report and documentation, have been uploaded to the GitHub repository.

### Repository Structure:

```
EcoFurnish-Repository/  
|-- src/  
|   |-- components/  
|   |-- app/  
|   |-- public/  
|   |-- styles/  
|-- .docs/  
|   |-- testing_report.csv  
|-- README.md
```

### README Summary:

- Instructions on how to run and test the application.
  - Details about implemented optimizations.
  - A link to the detailed testing report.
-