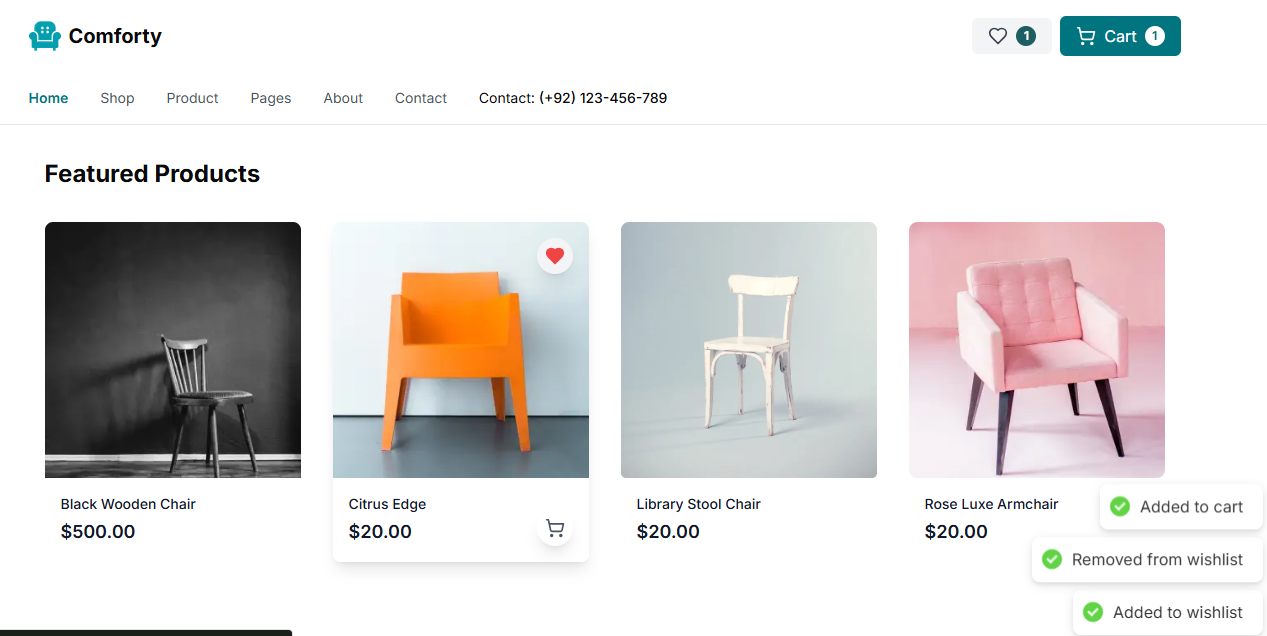
**Day 5 – Testing and Backend Refinement – Comforty**

**Overview:**

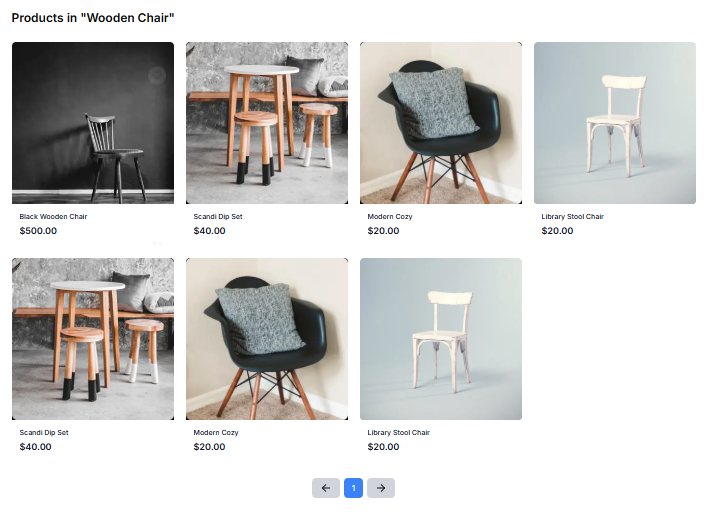
Comforty is a fully functional e-commerce platform focused on selling chairs, developed using Next.js, Sanity, and Tailwind CSS. It offers a seamless shopping experience with features like a product catalog, detailed product pages, shopping cart, and secure checkout. The platform is responsive, ensuring smooth performance across devices. With an intuitive interface and fast load times, Comforty delivers a modern, user-friendly solution for purchasing chairs online.

* **Functional Deliverables:**
* ***Functional & Responsive Components:***

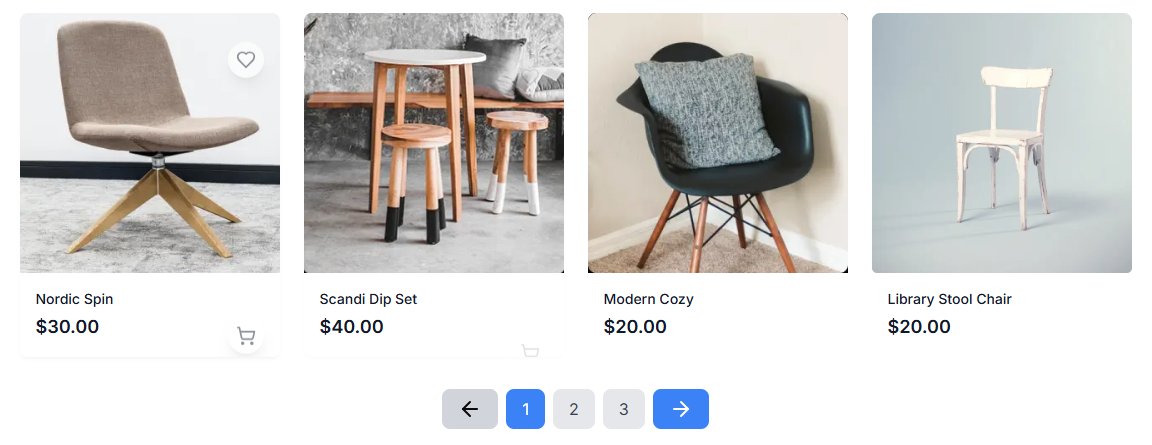
During the hackathon, I tested my e-commerce project using the Lighthouse extension and achieved 99% overall performance. The project features functional and responsive components, including add-to-cart functionality, notifications, and a Wishlist system, all designed to enhance the user experience.

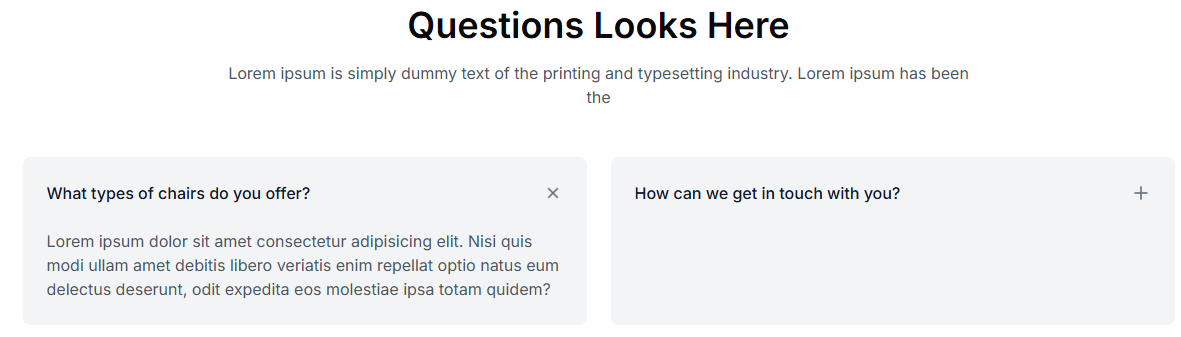


The category filter allows users to sort products by category, improving navigation and user experience. It dynamically updates the product list based on selected categories from your Sanity CMS.

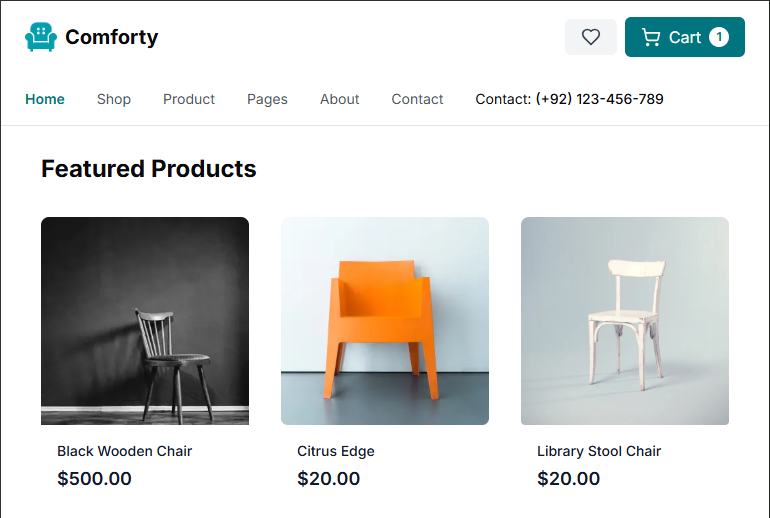
****

Pagination on the products page improves navigation by displaying a limited number of products per page. It uses Sanity CMS queries to fetch products dynamically based on the current page, reducing load times and enhancing the user experience.

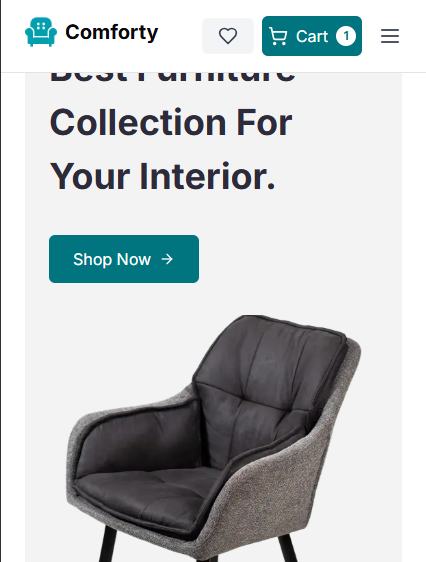
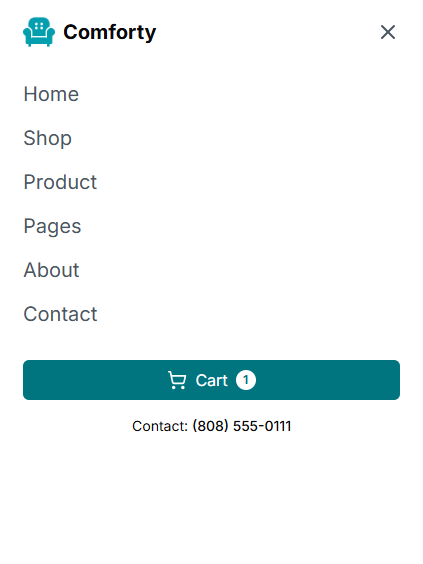
****

****

* **Responsive Screen (Tablet 768x564px):**

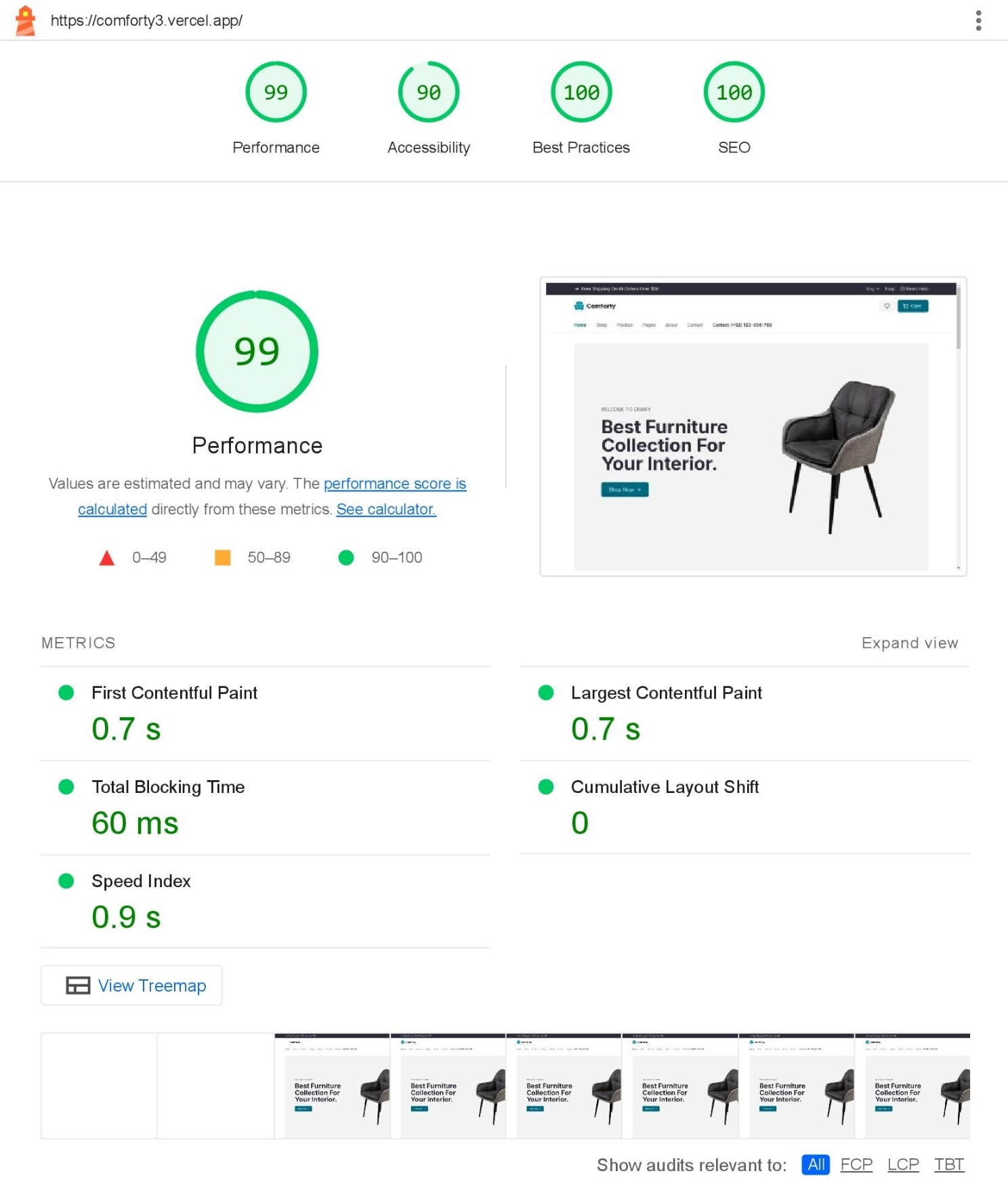
****

* **Responsive Screen (Mobile 768x564px):**

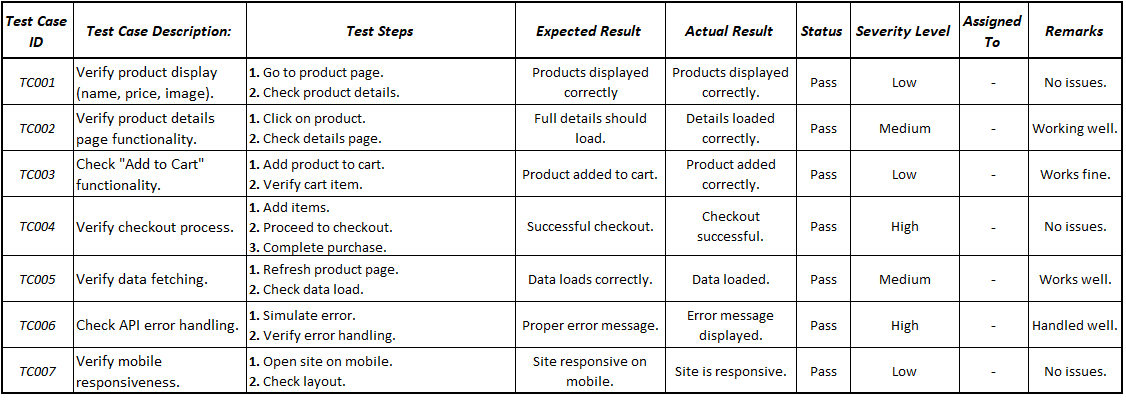
****

* ***Report from Testing Tool (Lighthouse):***

I tested my e-commerce project using the Lighthouse extension, achieving 99% overall performance. The only issue was a 1% slowdown caused by the integration of Snipcart, a third-party tool, due to its additional scripts.

****Despite this minor impact, Snipcart was essential for providing a seamless checkout experience. The project performed exceptionally well, showcasing the strength of the tech stack (Next.js, Sanity CMS, and Snipcart). Further optimization of third-party tools will be explored.

* **Testing Report (CSV Format):**

****

* **Documentation:**
* ***Report Detailing:***
* *Test Cases Executed & Results:*

All the test cases were executed successfully with the following outcomes:

* **Product Display Test**: Passed, all products were displayed correctly.
* **Product Detail Page Test**: Passed, product details loaded without issues.
* **Add to Cart Test**: Passed, products were added to the cart as expected.
* **Checkout Test**: Passed, the checkout process completed successfully.
* **Data Fetching Test**: Passed, data was fetched correctly without errors.
* **API Error Handling Test**: Passed, errors were handled with appropriate messages.
* **Mobile Responsiveness Test**: Passed, site worked well on mobile devices.
* *Performance Optimization:*

For performance optimization in a project using Next.js, Tailwind CSS, TypeScript, and Sanity:

1. **Code Splitting** to load only essential JavaScript.
2. **Tailwind CSS Purge** to remove unused styles.
3. **Next.js Image Optimization** for faster, responsive images.
4. **API Caching** with Sanity for faster data fetching.
5. **TypeScript** for type safety and fewer runtime errors.
6. **Optimized Font Loading** to reduce render-blocking.

* *Security measures implemented:*
* **HTTPS**: Enforcing secure connections to protect data in transit.
* **Environment Variables**: Storing sensitive data like API keys securely using environment variables.
* **Sanitizing Input**: Preventing injection attacks by sanitizing user inputs in forms and APIs.
* **Sanity CMS Security**: Configuring role-based access control (RBAC) in Sanity to limit content editing access.
* *Challenges Faced & Resolutions Applied:*

During the hackathon, key challenges included API integration issues, slow product loading, and category filtering. Resolutions included debugging API endpoints, implementing pagination for faster load times, and using efficient queries and client-side state management for accurate filtering. Additionally, responsive design was ensured using Next.js.