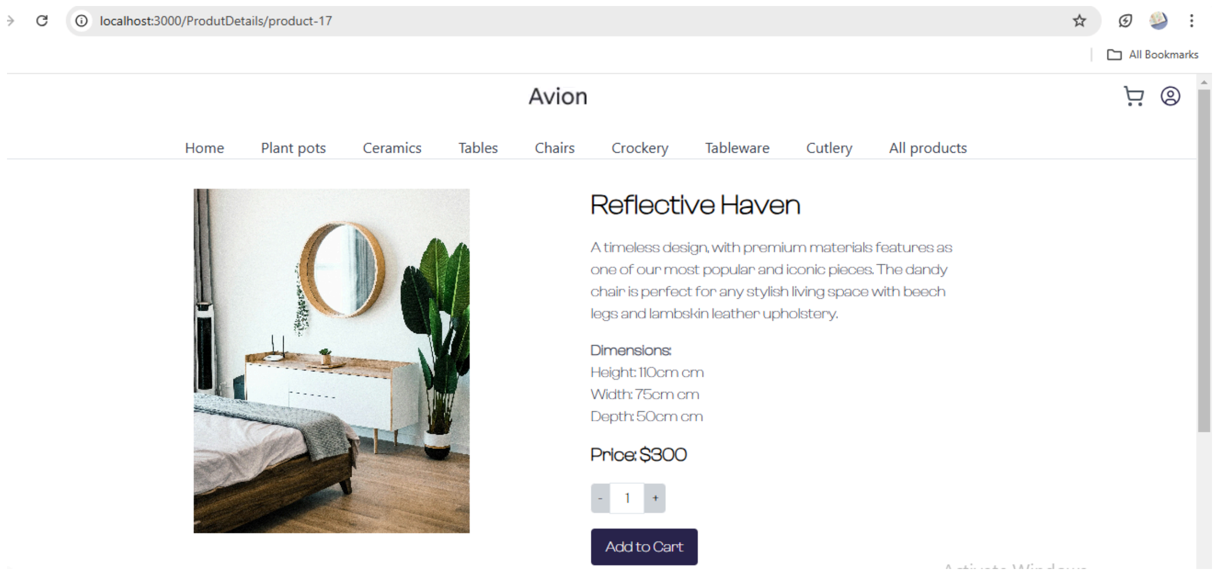
DAY-5 <u>Website Functionality and Feature Report</u>

1. Dynamic Routing Functionality

Overview:
The website utilizes dynamic routing to enable product-specific pages. This allows each product to have its own unique URL, making navigation more intuitive and personalized.

Implementation Details:
The routing system is managed by Next.js, which leverages dynamic routes to generate product pages. Each product is identified by its unique ID in the URL (e.g., /product/[id]). When a user visits a product's URL, the corresponding product data is fetched dynamically from the backend and rendered on the page.



Reflective Haven

A timeless design, with premium materials features as one of our most popular and iconic pieces. The dandy chair is perfect for any stylish living space with beech legs and lambskin leather upholstery.

Dimensions:
Height: 110cm cm
Width: 75cm cm
Depth: 50cm cm

Price: $300

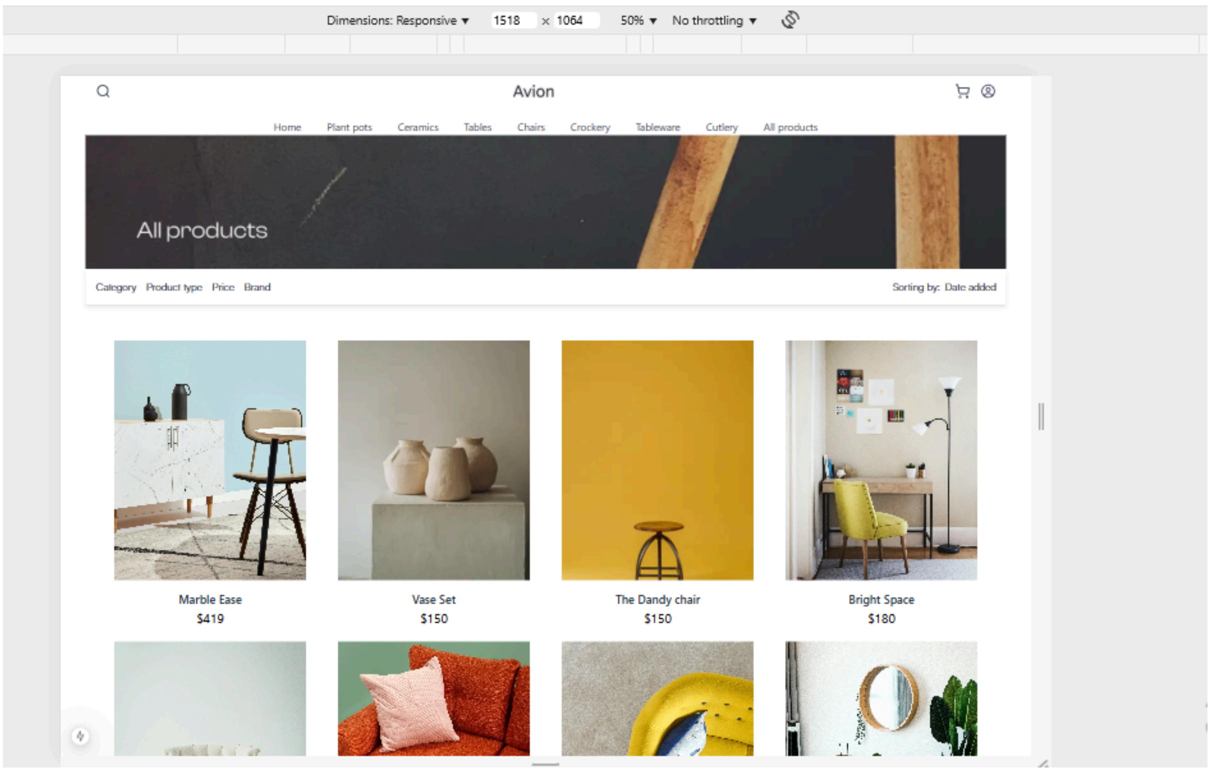
2. Product Listing

Overview:
The website displays a comprehensive list of available products. Each product is presented with relevant information such as product name, image, and price.

Implementation Details:

The product list is fetched from an Sanity and displayed on a designated page (e.g., the homepage or a products page).

Products are displayed in a card format with a link to each product's detail page.

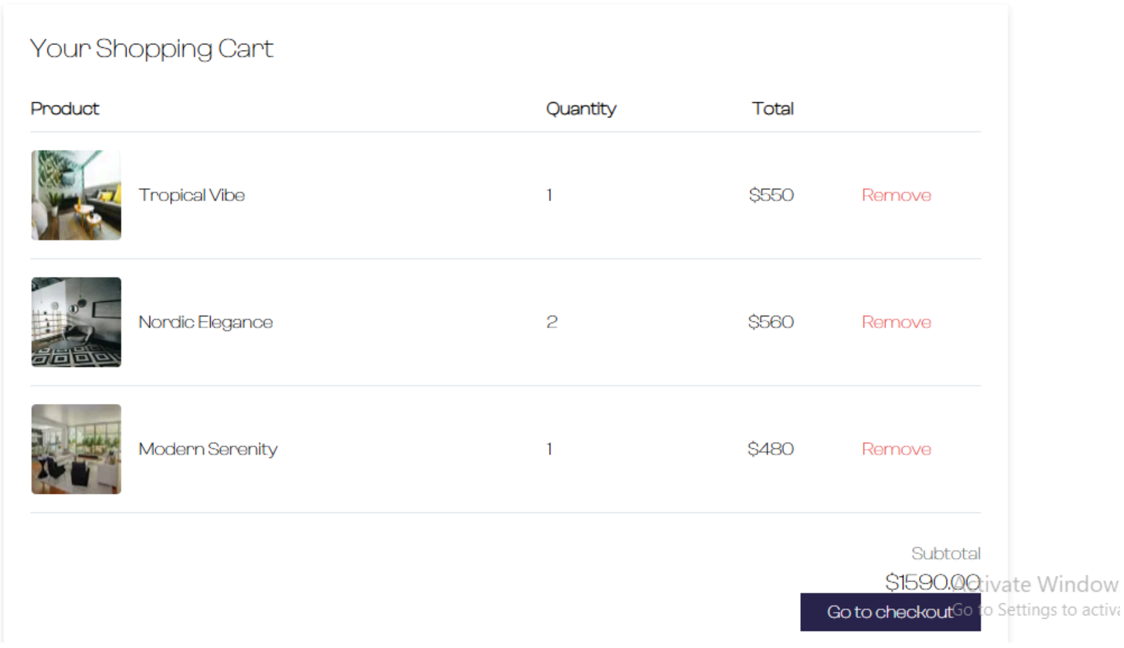


3. Cart Functionality

Overview:
The shopping cart allows users to add, remove, and modify quantities of products. The cart persists across different pages and even after refreshing the browser.

Implementation Details:

When a user clicks the "Add to Cart" button, the product is added to the cart stored in the browser's local storage or session.
The cart page allows users to review the contents, adjust quantities, and remove items.
The total price is dynamically calculated based on the cart's contents



Your Shopping Cart

Product	Quantity	Total	
Tropical Vibe	1	$550	Remove
Nordic Elegance	2	$560	Remove
Modern Serenity	1	$480	Remove

Subtotal

$1590.00

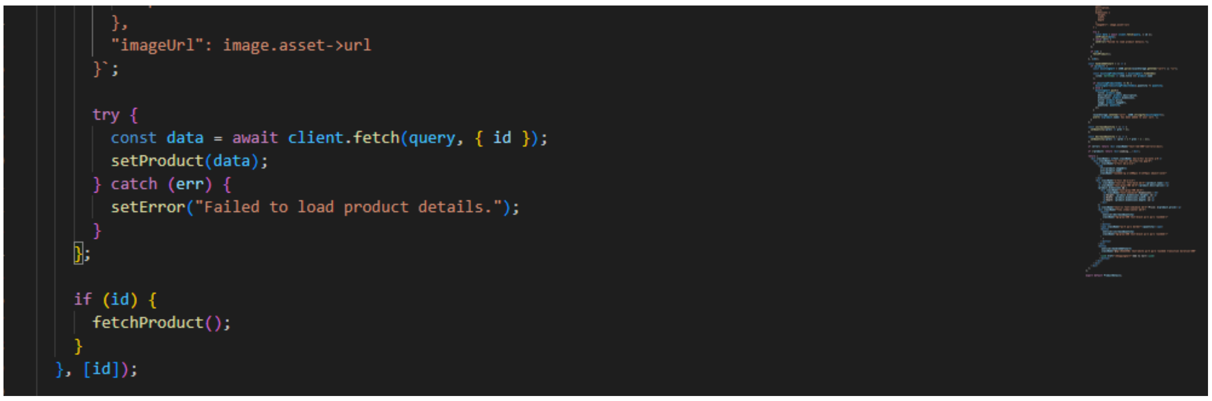
.

4. Error Handling

Overview:
The website implements error handling mechanisms to gracefully handle issues such as API failures or network errors.

Implementation Details:

Try-Catch Blocks: Used to handle API errors. If an error occurs while fetching product details or performing any other operation, the error is caught, and a user-friendly message is displayed.
Fallback Messages: In case of failure to load product data or any unexpected issues, a fallback message (e.g., "Oops! Something went wrong. Please try again later.") is shown to the user.

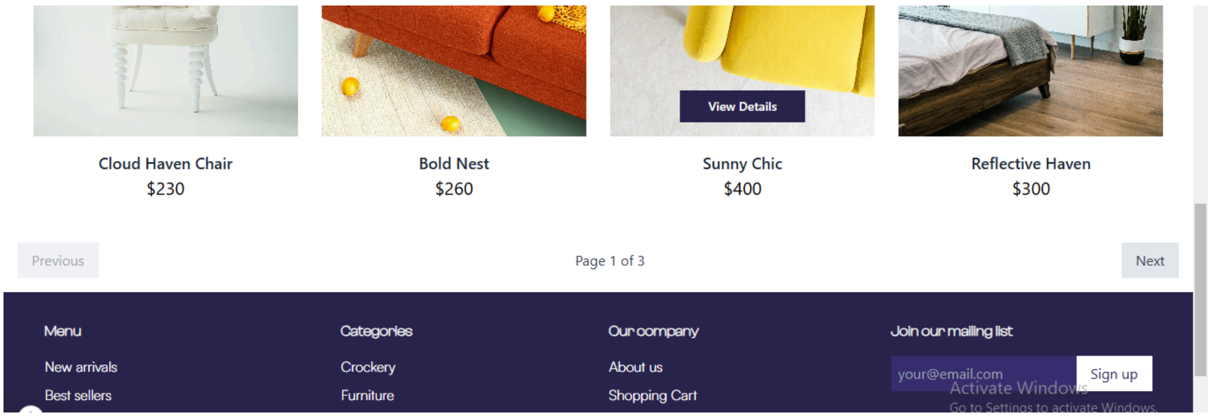


```
      },
      "imageUrl": image.asset->url
    }`;

    try {
      const data = await client.fetch(query, { id });
      setProduct(data);
    } catch (err) {
      setError("Failed to load product details.");
    }
  };

  if (id) {
    fetchProduct();
  }
}, [id]);
```

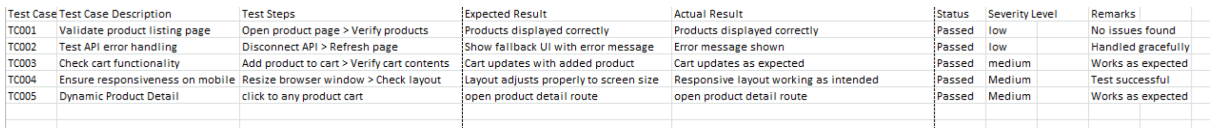
5. Pagination

Overview:

Pagination is implemented to improve the user experience by dividing the product list into manageable pages. This prevents overwhelming users with too much information at once.



6. Testing Report (CVS Format):

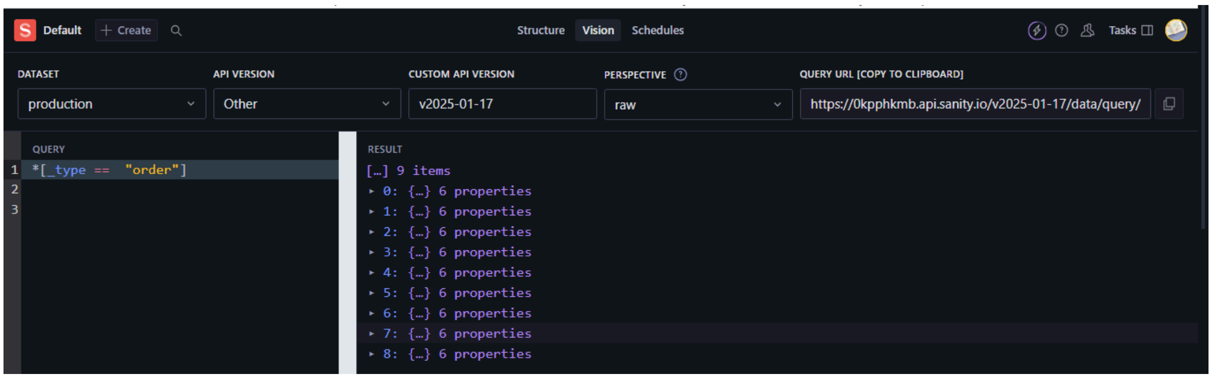


Test Case	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Validate product listing page	Open product page > Verify products	Products displayed correctly	Products displayed correctly	Passed	low	No issues found
TC002	Test API error handling	Disconnect API > Refresh page	Show fallback UI with error message	Error message shown	Passed	low	Handled gracefully
TC003	Check cart functionality	Add product to cart > Verify cart contents	Cart updates with added product	Cart updates as expected	Passed	medium	Works as expected
TC004	Ensure responsiveness on mobile	Resize browser window > Check layout	Layout adjusts properly to screen size	Responsive layout working as intended	Passed	Medium	Test successful
TC005	Dynamic Product Detail	click to any product cart	open product detail route	open product detail route	Passed	Medium	Works as expected

7. Data Storage in Sanity

Overview:

Sanity is used as the content management system (CMS) and database for managing product data. This ensures scalability and ease of content updates.



DATASET: production
API VERSION: Other
CUSTOM API VERSION: v2025-01-17
PERSPECTIVE: raw
QUERY URL [COPY TO CLIPBOARD]: https://0kpphkmb.api.sanity.io/v2025-01-17/data/query/

QUERY

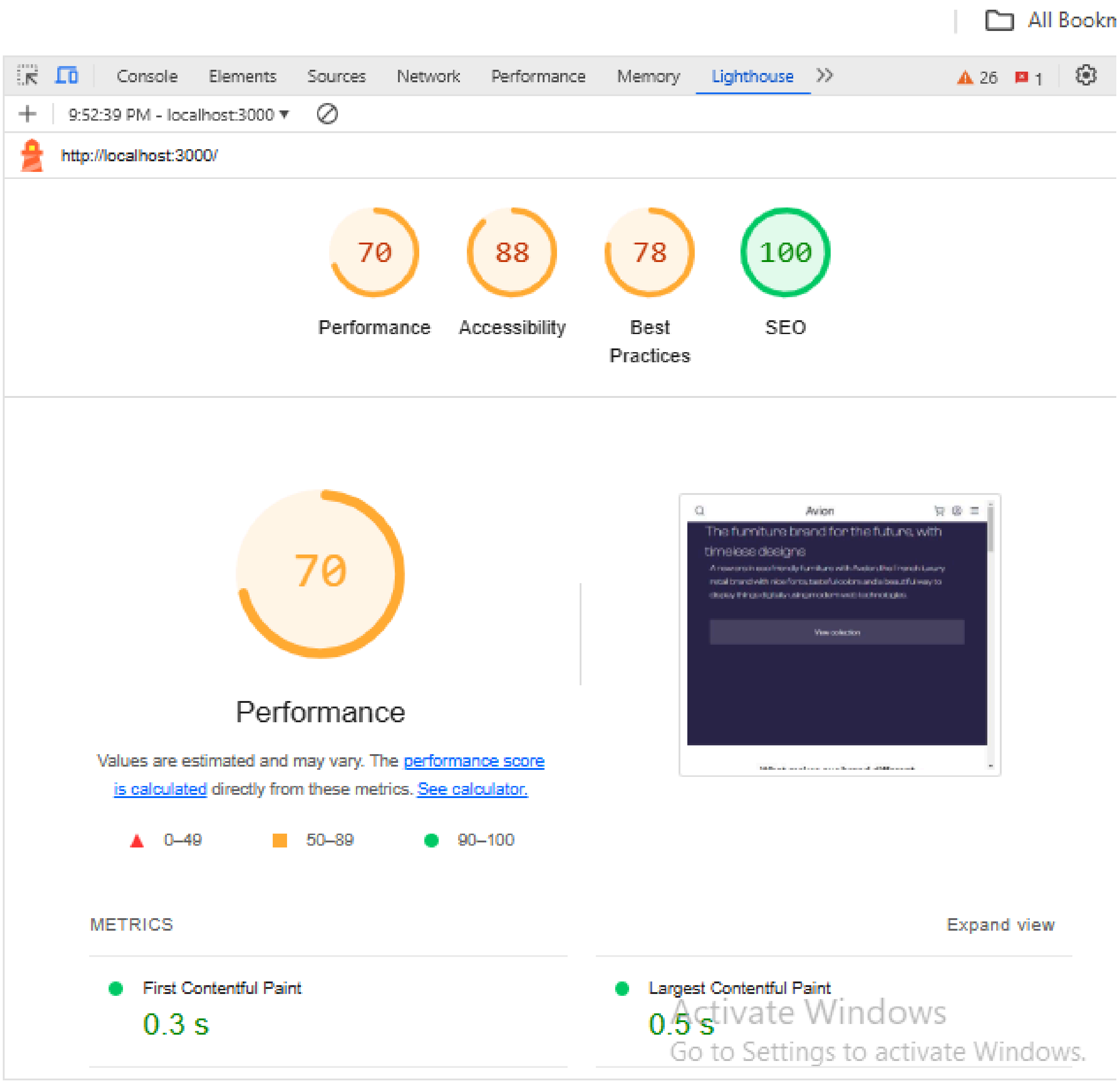
```
*[_type == "order"]
```

RESULT

```
[…] 9 items
▸ 0: {…} 6 properties
▸ 1: {…} 6 properties
▸ 2: {…} 6 properties
▸ 3: {…} 6 properties
▸ 4: {…} 6 properties
▸ 5: {…} 6 properties
▸ 6: {…} 6 properties
▸ 7: {…} 6 properties
▸ 8: {…} 6 properties
```

I am focused on improving order management in Sanity to ensure it is more reliable,and perfectly aligned with the website's needs

8. .Analyze Performance:


All Bookn

Console Elements Sources Network Performance Memory Lighthouse

9:52:39 PM - localhost:3000

http://localhost:3000/

70 Performance
88 Accessibility
78 Best Practices
100 SEO

70
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
0.3 s

● Largest Contentful Paint
0.5 s

Activate Windows
Go to Settings to activate Windows.

Conclusion

The website's core functionalities, such as dynamic routing, product listing, pagination, cart operations, error handling, and performance optimization, are designed to offer a smooth and reliable user experience. Currently, I am actively working on refining these features to make them even more efficient and user-friendly.

prepared by: Hafsa Sagheer Ahmed (Senior Student)

Date: 21-1-2025