In day5 I focus mainly on understanding Cypress and write test cases for this

- Add to cart
  - describe('Cart Operations - Add Items by Product ID', () => {
  - beforeEach(() => {
  - // Navigate directly to the product details page using a hardcoded product ID
  - const productId = 'YjFIo1g1LQZHJZg27PUDNv'; // Replace with a valid product ID
  - cy.visit(`/product/${productId}`);
  - });
  -
  - it('should add the product to the cart from the product details page', () => {
  - // Verify the product details are displayed
  - cy.get('[data-testid="product-title"]').should('exist'); // Replace with the actual product name
  - cy.get('[data-testid="product-price"]').should('exist');
  -
  - // Add the product to the cart
  - cy.get('[data-testid="add-to-cart"]').click(); // Add the product to the cart
  - cy.get('[data-testid="cart-icon"]').click();
  - // Open the cart and verify the product is added
  - cy.get('[data-testid="cart-item"]').should('have.length', 1);
  - cy.get('[data-testid="cart-item"]').should('contain',); // Verify the correct product is added
  - });
  - });
- Update cart
  - describe('Cart Operations - Update Items', () => {
  - const productId = 'YjFIo1g1LQZHJZg27PUDNv'; // Replace with a valid product ID
  - beforeEach(() => {
  - // Navigate directly to the product details page using a hardcoded product ID
  - cy.visit(`/product/${productId}`);
  - cy.get('[data-testid="add-to-cart"]').click();
  - });
  - cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
  - cy.get('[data-testid="quantity-input"]').should('contain', '1'); // Check initial quantity
  - });
  -
  - // Increase the quantity
  - cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
  - cy.get('[data-testid="quantity-increase"]').click(); // Click the + button
  - });
  -

- // Verify the updated quantity after increment
- cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
- cy.get('[data-testid="quantity-input"]').should('contain', '2'); // Verify quantity incremented
- });
-
- // Decrease the quantity
- cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
- cy.get('[data-testid="quantity-decrease"]').click(); // Click the - button
- });
-
- // Verify the updated quantity after decrement
- cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
- cy.get('[data-testid="quantity-input"]').should('contain', '1'); // Verify quantity decremented
- });
-
- // Ensure the quantity does not go below 1
- cy.get(`[data-testid="cart-item-${productId}"]`).within(() => {
- cy.get('[data-testid="quantity-decrease"]').click(); // Try to decrease below 1
- cy.get('[data-testid="quantity-input"]').should('contain', '1'); // Verify it remains at 1
- });
- });
-

- Home page
  - Cypress code
    - describe("Navigation Tests", () => {
    - it("should load the homepage", () => {
    - // Visit the homepage
    - cy.visit("/");
    -
    - // Verify that the URL is correct
    - cy.url().should("eq", "http://localhost:3000/");
    -
    - // Check if a unique element like the logo, header, or navbar exists
    - });
    -
    - it("should navigate to the About page", () => {
    - // Visit the homepage
    - cy.visit("/");
    -
    - // Click on the About page link
    - cy.get('a[href="/about"]').click();
    -
    - // Verify the URL has changed to /about

```
    cy.url().should("include", "/about");

    // Check if the About page has a unique element or identifier
    });
});
```

- Dynamic Routing

```
describe('Dynamic Routing Tests', () => {
  const baseUrl = 'http://localhost:3000'; // Your app's base URL
  const categoryId = '0e0ztuJY6NWdyblBES3JSp';
  const productId = 'YjFIo1g1LQZHJZg27PUDNv';

  it('should navigate to the category page with a dynamic ID', () => {
    // Visit the category page
    cy.visit(`${baseUrl}/category/${categoryId}`);

    // Assert the correct category page loads
    cy.url().should('include', `/category/${categoryId}`);
    cy.get('h1').should('exist'); // Adjust to check for specific content, e.g., category name
  });

  it('should navigate to the product page with a dynamic ID', () => {
    // Visit the product page
    cy.visit(`${baseUrl}/product/${productId}`);

    // Assert the correct product page loads
    cy.url().should('include', `/product/${productId}`);
    cy.get('h1').should('exist'); // Adjust to check for specific product title
  });
 });
```

- Product categories components

```
describe('Product Categories Component', () => {
  beforeEach(() => {
    cy.visit('/'); // Navigate to the homepage where Products component is located
  });

  it('should display product categories with images', () => {
    // Check if the "Top Categories" title is present
    cy.get('h1').contains('Top Categories');

    // Check if the slider is visible
    cy.get('.slider-container').should('be.visible');

    // Verify that categories are displayed with images
```

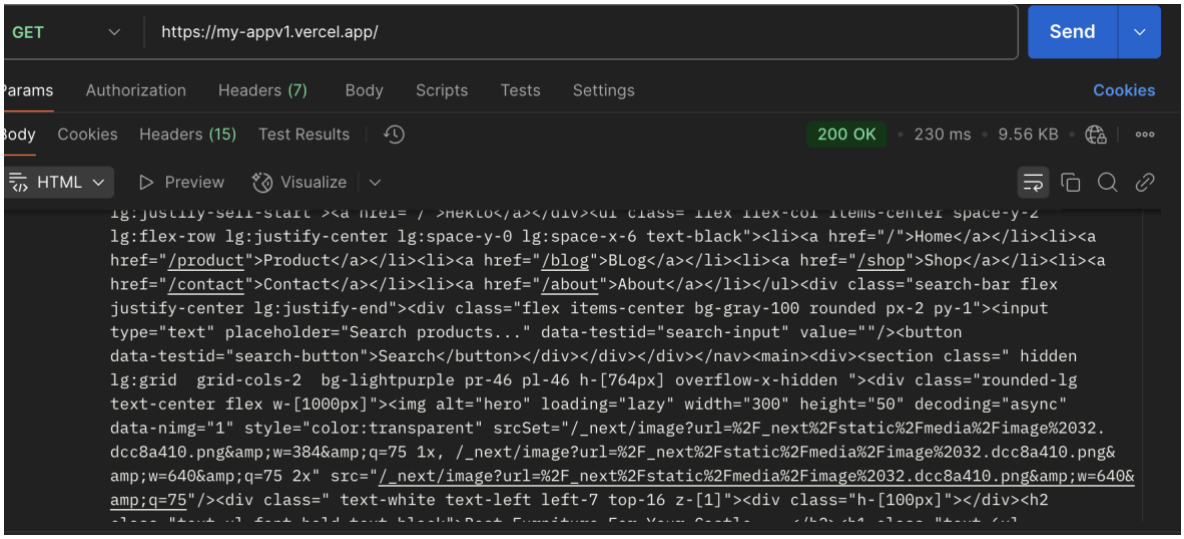
```
      cy.get('[data-testid="product-category"]').should('have.length.greaterThan', 0); // Check for categories

      // Check that each category has an image and a name
      cy.get('[data-testid="product-category"]').each(($category) => {
        cy.wrap($category).find('img').should('be.visible'); // Ensure image is present
        cy.wrap($category).find('h3').should('not.be.empty'); // Ensure name is present
      });
    });

    it('should navigate to category page when "Read More" is clicked', () => {
      // Click on the "Read More" link for the first category
      cy.get('[data-testid="product-category"]')
        .first()
        .find('a')
        .click();

      // Ensure the navigation leads to the correct category page
      cy.url().should('include', '/category/');
    });
  });
```

- Search bar

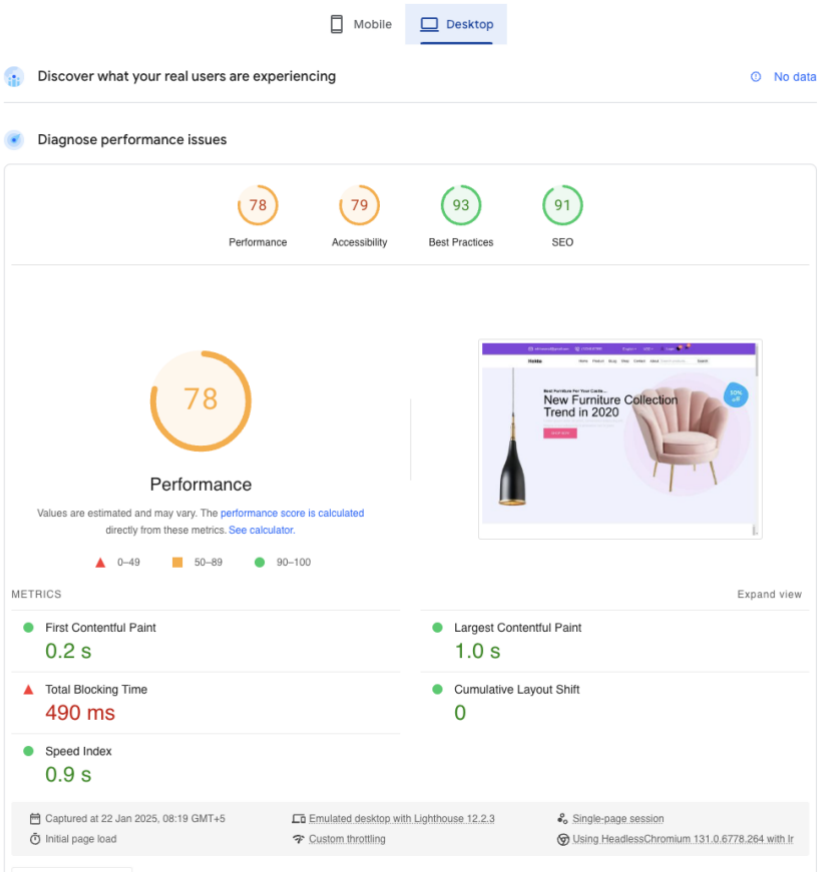
```
describe('Navbar Search Functionality', () => {
  beforeEach(() => {
    // Visit the page that contains the search functionality (replace with actual URL)
    cy.visit('/'); // Replace with the actual URL where the search bar exists
  });

  it('should fetch and display products based on the search query', () => {
    const searchQuery = 'chair'; // Define a sample search query

    // Type into the search input field
    cy.get('[data-testid="search-input"]').type(searchQuery); // Adjust selector for your search input field

    // Click the search button to trigger the search
    cy.get('[data-testid="search-button"]').click(); // Adjust selector for your search button
```

```
    // Wait for the products to be fetched (add a wait or check network request if necessary)
    cy.wait(500); // Optional: Add wait if there's a delay in fetching results

    // Verify that the products are displayed based on the search query
    cy.get('[data-testid="search-result"]').each(($el) => {
      // Verify that each result contains the search term
      cy.wrap($el).should('contain.text', searchQuery);
    });

    // Optionally, verify that the search query is reflected in the results section (e.g., title or header)
    cy.get('[data-testid="search-results-header"]').should('contain.text', `Results for "${searchQuery}"`);
  });

  it('should display a no results message when no products match the search query', () => {
    const noResultsQuery = 'NonExistentProduct'; // Define a search query with no matching products

    // Type into the search input field
    cy.get('[data-testid="search-input"]').type(noResultsQuery);

    // Click the search button
    cy.get('[data-testid="search-button"]').click();

    // Wait for results to be fetched
    cy.wait(500);

    // Verify that no products are displayed
    cy.get('[data-testid="search-result"]').should('have.length', 0);

    // Optionally, verify that a "no results" message is displayed
    cy.get('[data-testid="no-results-message"]').should('be.visible').and('contain.text', 'No results found');
  });
});
```

Postman



Performance checking



-

DIAGNOSTICS

- ▲ Avoid an excessive DOM size — 3,021 elements
- ▲ Largest contentful paint image was lazily loaded
- ■ Ensure text remains visible during webfont load
- ■ Avoid serving legacy JavaScript to modern browsers — Potential savings of 0 KiB
- ■ Reduce unused JavaScript — Potential savings of 21 KiB
- ○ Minimises main-thread work — 1.7 s
- ○ Avoid long main-thread tasks — 3 long tasks found
- ○ JavaScript execution time — 1.0 s
- ○ Initial server response time was short — Root document took 10 ms
- ○ Avoids enormous network payloads — Total size was 499 KiB
- ○ Avoid chaining critical requests — 2 chains found
- ○ Minimise third-party usage — Third-party code blocked the main thread for 0 ms
- ○ Largest contentful paint element — 1,000 ms

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

PASSED AUDITS (23) Show

PASSED AUDITS (23) Show

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.

ARIA

- ▲ `[aria-hidden="true"]` elements contain focusable descendents

These are opportunities to improve the usage of ARIA in your application which may enhance the experience for users of assistive technology, such as a screen reader.

CONTRAST

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

These are opportunities to improve the legibility of your content.

TABLES AND LISTS

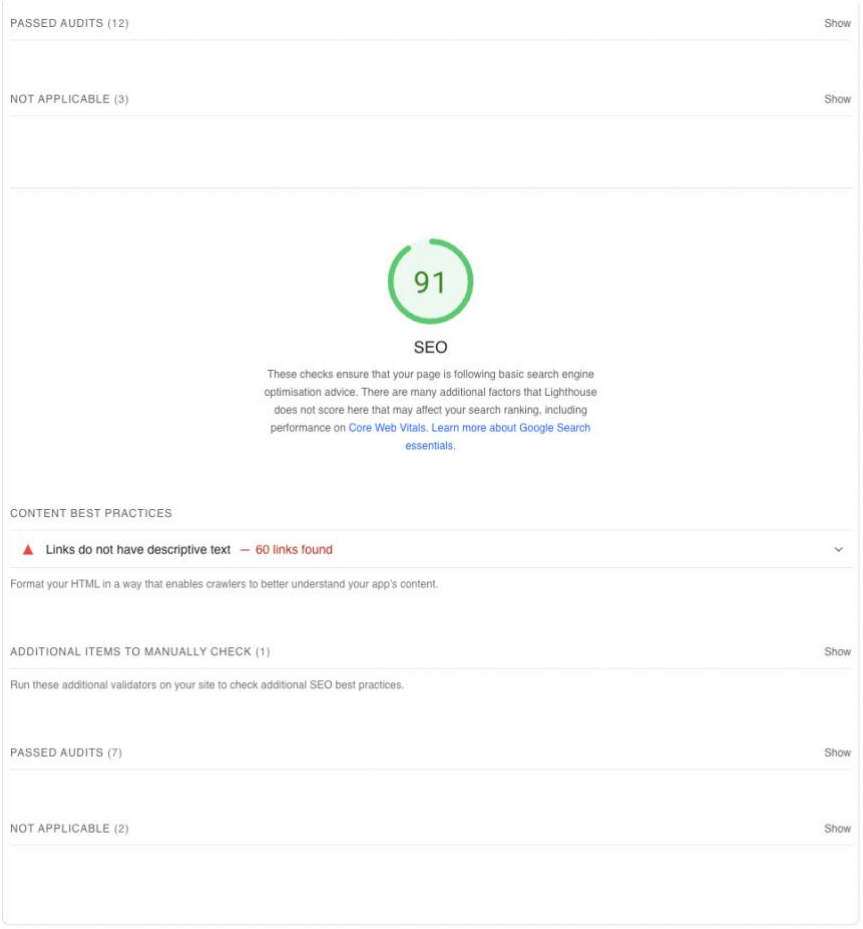
- ▲ Lists do not contain only `<li>` elements and script supporting elements (`<script>` and `<template>`).
- ▲ List items (`<li>`) are not contained within `<ul>`, `<ol>` or `<menu>` parent elements.

These are opportunities to improve the experience of reading tabular or list data using assistive technology, like a screen reader.

NAMES AND LABELS

- ▲ Select elements do not have associated label elements.

These are opportunities to improve the semantics of the controls in your application. This may enhance the experience for users of assistive technology, such as a screen reader.



Error handling images

```
{c.image ? (
                  <Image
                    src={urlFor(c.image).url()} // Use category.image
                    alt={c.name || "Category Image"}
                    width={200}
                    height={200}
                  />
                ) : (
                  <p>No image available</p>
                )}
```

Error handling in product ->[id]->page.tsx

```
useEffect(() => {
      const fetchProductData = async () => {
        try {
          const query = `*[_type == "product" && _id == $productId]{
            _id,
            name,
```

```
      description,
      price,
      image,
      category-> { _id, name }
    }[0]`;
    const data = await client.fetch(query, { productId: params.id });
    setProduct(data);
  } catch (err) {
    setError("Failed to fetch product data");
  } finally {
    setIsLoading(false);
  }
};

fetchProductData();
}, [params.id]);
```

Using burpesuit is paid so we have proxy 8080 and put the app in url that feature was disable for free version.

Cypress results