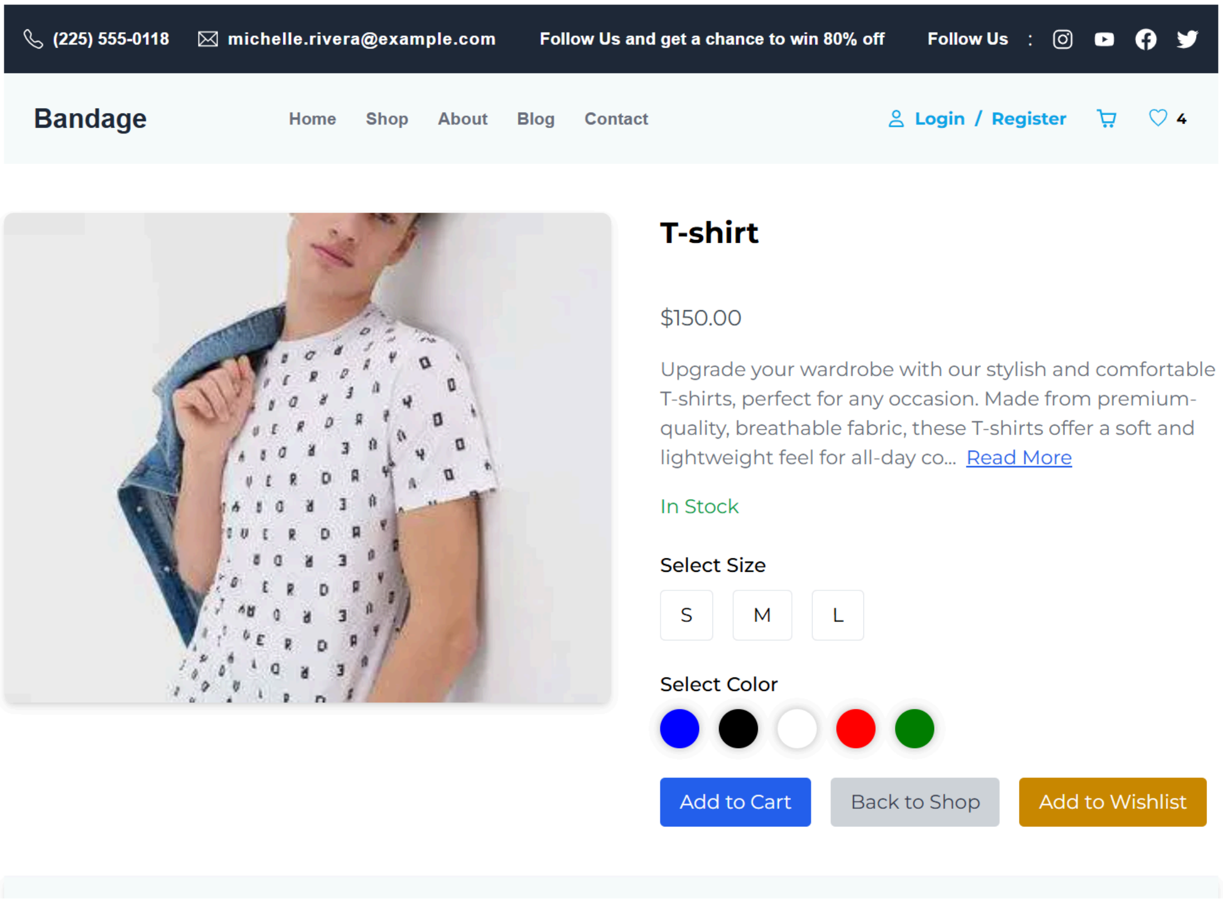
Day 5 - Testing and Backend Refinement - [General Cloth E-Commerce]

1) <u>Product page, Search bar, Add to Cart functionality, Checkout page:</u>

1) <u>Desktop View:</u>



(225) 555-0118 michelle.rivera@example.com Follow Us and get a chance to win 80% off Follow Us :

Bandage Home Shop About Blog Contact Login / Register 4

T-shirt

$150.00

Upgrade your wardrobe with our stylish and comfortable T-shirts, perfect for any occasion. Made from premium-quality, breathable fabric, these T-shirts offer a soft and lightweight feel for all-day co... Read More

In Stock

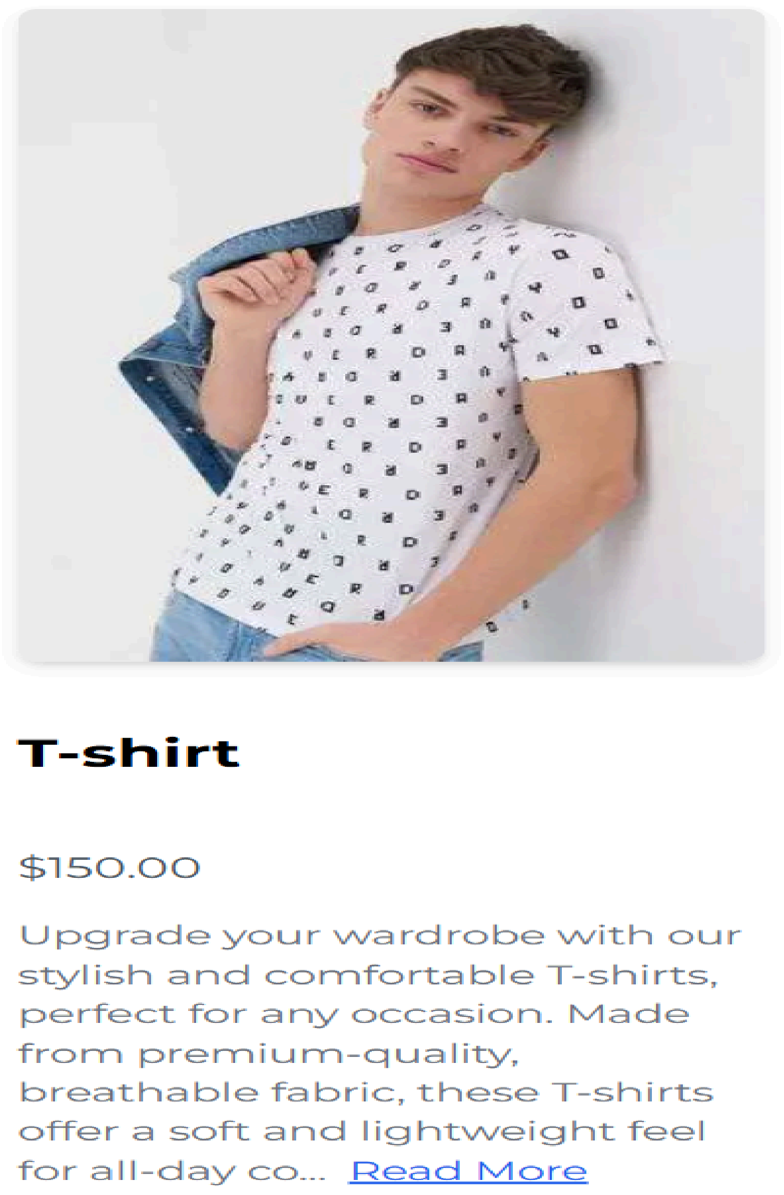
Select Size

S M L

Select Color

Add to Cart Back to Shop Add to Wishlist

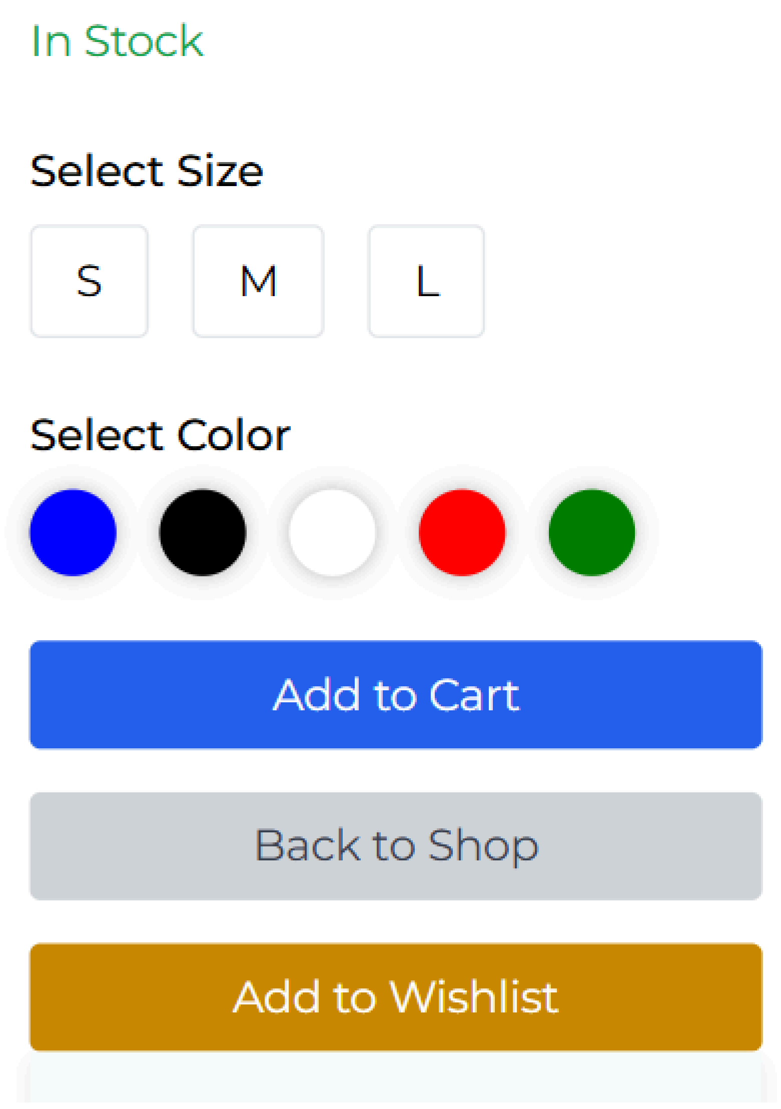
2) **<u>Mobile View:</u>**



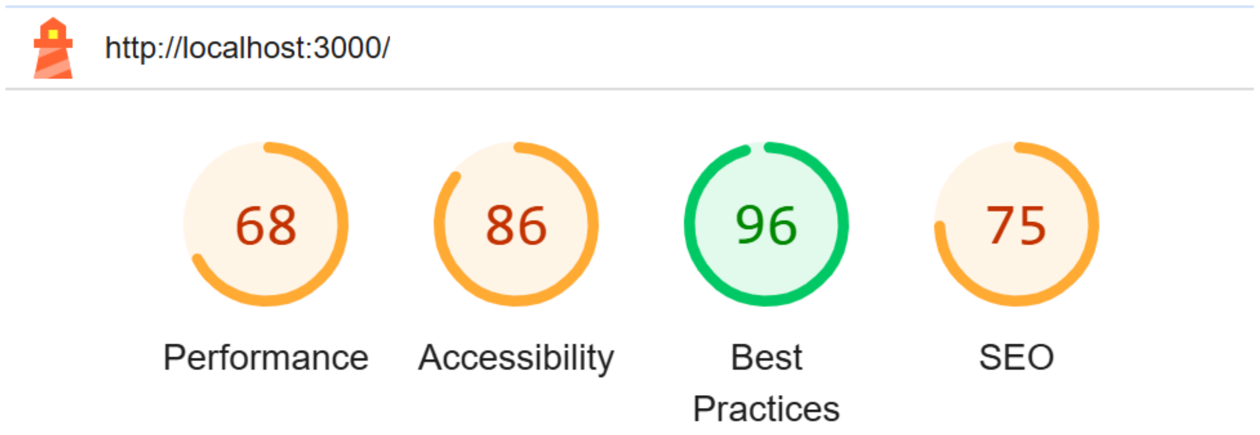
T-shirt

$150.00

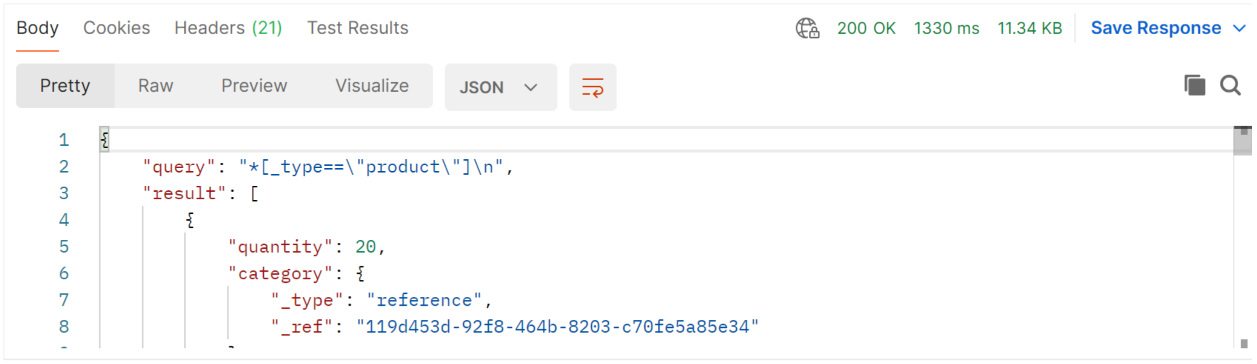
Upgrade your wardrobe with our stylish and comfortable T-shirts, perfect for any occasion. Made from premium-quality, breathable fabric, these T-shirts offer a soft and lightweight feel for all-day co... Read More



Lighthouse Report:



Postman Report:



```
{
    "query": "*[_type==\"product\"]\n",
    "result": [
        {
            "quantity": 20,
            "category": {
                "_type": "reference",
                "_ref": "119d453d-92f8-464b-8203-c70fe5a85e34"
```

Testing Report (CSV Format):

Test Case ID	**Test Case Description**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Severity Level**	**Remarks**
TC001	"Add to Cart" Feature Test	1. Go to the product page. 2. Click on the "Add to Cart" button. 3. Go to the cart page to verify the product has been added to your Cart.	Product will appear in the cart.	Product appeared in the cart as expected.	Passed	Low	All steps followed correctly.
TC002	Checkout Process Test	1. Add product to cart. 2. Proceed to checkout. 3. Click "Place Order".	Order will be placed successfull y and confirmatio n page Will appear.	Order was placed and confirmation page appeared.	Passed	Medium	No errors in the checkout process.
TC003	Search Functionality Test	1. Go to Shop page. 2. Type a product name in the search bar. 3. Click on the search icon.	Relevant products will be displayed based on the search term.	Relevant products appeared in the search results.	Passed	Low	Search is working fine.
TC004	Wishlist Feature Test	1. Go to product page. 2. Click on "Add to	Product will be added to		Passed	Low	Wishlist is functionin

		Wishlist" button. 3. Go to the wishlist page to verify the item is added.	the wishlist.	Product successfully added to the wishlist.			g properly.
TC005	Mobile Responsiveness Test	1. Open website on mobile device. 2. Check if all components are responsive and properly aligned.	Website will be fully responsive on mobile devices.	Website is fully responsive on mobile.	Passed	Medium	No responsiveness issues found.

1) Testing Approach:

- Manual Testing: Executed functional tests by interacting with the system.
- Automated Testing: Used Lighthouse for performance audits and Postman for API testing.
- Security Testing: Checked for HTTPS usage and secure API communications.

Key Findings

- Performance Issues: CSS minification needed for better load times.
- Security Enhancements: Ensure all API requests are made over HTTPS.
- Functional Bugs: No major functional bugs found.

Performance Optimization Steps Taken:

1. CSS & JavaScript Minification

- Enabled SWC Minification in `next.config.js` to optimize JavaScript bundle size.

- Removed unused CSS classes by purging Tailwind CSS using `content` paths in `tailwind.config.ts`.
- Used JIT mode in Tailwind CSS for faster build and reduced CSS file size.

2. Image Optimization

- Used **Next.js Image Component (`next/image`)** for automatic image optimization.
- **Set proper dimensions & quality** in images to balance performance and visual quality.
- Configured **Sanity CMS images** with `remotePatterns` in `next.config.js` to allow optimized image fetching.

4. Code Splitting & Lazy Loading

- Implemented **dynamic imports (`next/dynamic`)** for components that are not needed on initial page load.
- Used **React Suspense & Lazy Loading** to defer loading non-critical components.

Security Measures Implemented:

1. Secure API Communication

- All API calls are made over HTTPS to prevent data interception.
- Implemented CORS policy to restrict API access from unauthorized domains.
- Used Sanity API tokens with proper permissions to secure backend access.

3. Data Validation & Input Sanitization

- Used **Zod for schema validation** to prevent incorrect or malicious data input.
- Sanitized user inputs to prevent **XSS (Cross-Site Scripting) attacks**.

5. Database Security

- Restricted direct database access and allowed only API-based interactions.
- Used **Sanity dataset security settings** to limit public read/write access.
- Implemented **rate limiting** to prevent brute-force attacks on APIs.

Challenges Faced and Resolutions Applied:

1. API Integration Issues

Challenge:

- Sanity CMS API response structure was complex and required adjustments in fetching logic.

Resolution:

- Optimized GROQ queries to fetch only required fields, reducing payload size.
- Implemented error handling for API failures and network issues.

2. Performance Optimization

Challenge:

- **Lighthouse audit** flagged issues like unoptimized images, render-blocking scripts, and large CSS files.

Resolution:

- Used **Next.js Image Optimization** to dynamically serve compressed images.
- Minified CSS using **Tailwind JIT mode** and enabled **SWC minification** in `next.config.js`.
- Lazy-loaded components to improve initial page load time.

6. Search Functionality Challenges

Challenge:

- Implementing **real-time search** using Sanity GROQ queries was complex.

Resolution:

- Optimized **search queries** and used **debouncing** to prevent excessive API calls.
- Implemented **client-side caching** to improve response time.

Conclusion:

During the process of testing and backend refinement, we implemented **functional testing, performance optimization, and security enhancements**. Using **Lighthouse, Postman, and manual testing**, we identified system flaws and resolved them effectively. To optimize performance, we applied **CSS minification, lazy loading, and efficient API calls**, while security was ensured through **HTTPS enforcement, authentication, and secure API communication**.

Name: Muneeb Jawed

Roll No. 00355255

Class: Sunday 2 to 5