Day 5 - Testing and Backend Refinement - [SHOP.CO]

1. Functional Deliverables

- **Functional and Responsive Components**:
- **Postman API Testing Logs**

2. Testing Report :

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC-001	Verify Product List Loads Correctly	1. Open the marketplace homepage. 2. Check the product list.	Products should display with correct information and images.	Products displayed correctly.	Passsed	Low	N/A
TC-002	Check API Integration for Product Data	1. Trigger API call to `/api/products`. 2. Check response.	Should return product data with no errors.	API responded correctly.	Passsed	High	API key needs to be secured.
TC-003	Test Responsive Design on Mobile	1. Open site on a mobile device. 2. Navigate through the pages.	All pages should be responsive and user-friendly on mobile.	Pages are responsive on mobile.	Passsed	Low	Minor layout adjustment needed.

TC-004	Test Error Handling for Missing Data	1. Simulate API failure. 2. Check UI for fallback.	A "No products available" message should appear.	Fallback UI shown correctly.	Passed	High	Ensure error handling is robust.

3. Documentation

Test Cases Executed and Their Results:

- **Functional Tests**:

All functional tests, including product listing display, search functionality, and product detail page routing, passed successfully.

- **Responsive Design Tests**:

Responsive testing was performed across multiple devices, ensuring that the layout adjusted correctly on mobile, tablet, and desktop views.

- **API Tests**:

API tests for fetching product data worked without issues. However, there were concerns about secure handling of API keys, which were addressed by storing them in environment variables.

Performance Optimization Steps Taken:

- **Performance Audits**:

Ran Lighthouse audits and identified areas for improvement, such as reducing unused JavaScript and CSS. These were addressed by optimizing and bundling the assets.

Security Measures Implemented:

- **Secure API Key Handling**:

All sensitive API keys were moved to environment variables to avoid exposure in the frontend code.

- **HTTPS Enforcement**:

Ensured that the application runs over HTTPS for secure communication.

Challenges Faced and Solutions Applied:

- **Challenge 1**: Handling API failures gracefully when the Sanity CMS API was down.

Solution: Implemented a fallback UI with error messages like "No products available."

- **Challenge 2**: Ensuring responsiveness across multiple devices.

Solution: Used BrowserStack and physical device testing to ensure compatibility.