

Day 5: Testing & Integration

Prepared by: Javeria Nigar

Project: E-commerce Marketplace – Nike

Overview & Objectives

The primary goal of Day 5 is to validate the functionality, performance, and security of the E-commerce marketplace before deployment. Key objectives include:

Comprehensive Testing: Functional, performance, security, and user acceptance testing.

Error Handling: Implementing clear error messages and fallback mechanisms.

Backend Integration Refinement: Ensuring seamless API communication and data flow.

Performance Optimization: Enhancing page load speed and responsiveness.

Security Enhancements: Safeguarding user data and API keys from potential threats.

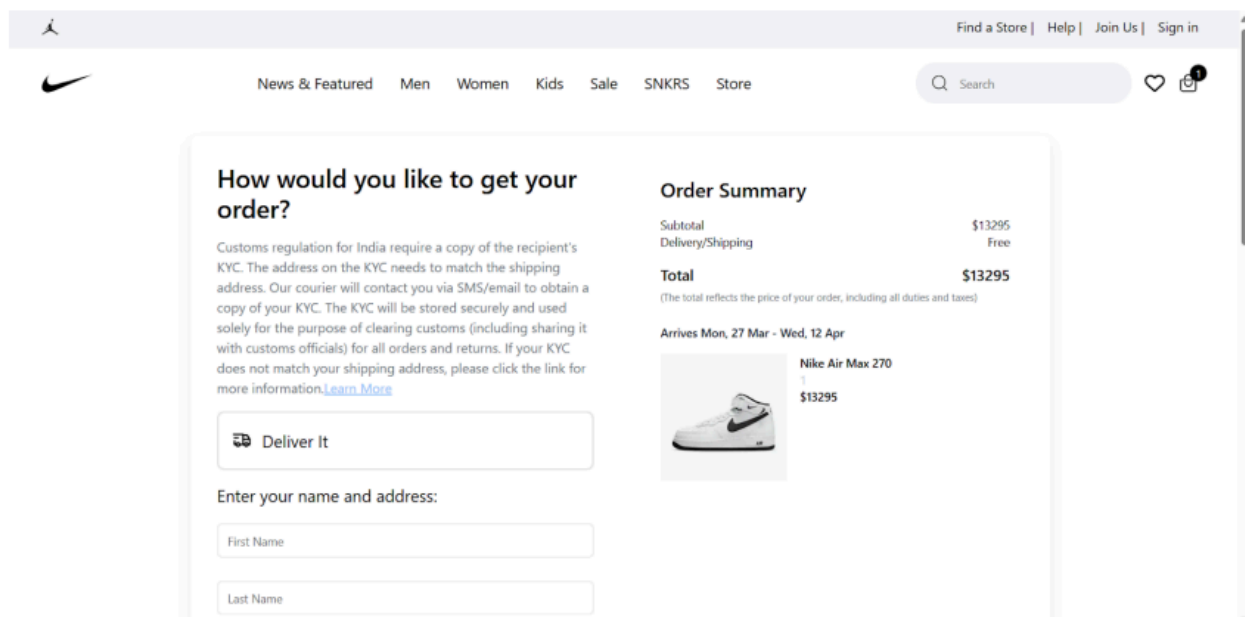
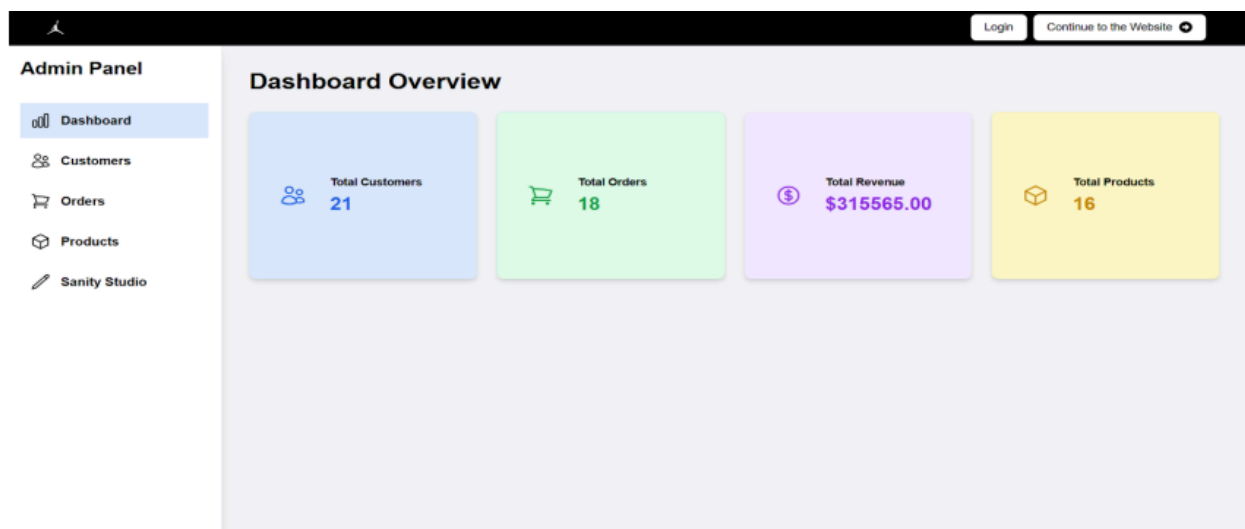
Implementation & Testing Process

1 Functional Testing

Testing was conducted on the following components:

- ✓ Product Listing
- ✓ Categories
- ✓ Best-Selling Products
- ✓ Header & Footer
- ✓ Featured Products (Fetched from Sanity CMS)
- ✓ Help Page

- ✓ Authentication (Login & Register via Clerk)
- ✓ Dynamic Product Pages (Slug-based Routing)
- ✓ Add to Cart (With Toast Notifications)
- ✓ Checkout & Order Submission (Toast Notifications)
- ✓ Order Data Storage in Sanity CMS
- ✓ Admin Dashboard (Restricted Access)
- ✓ Orders & Products Management (Admin Panel)
- ✓ Customer Data Storage in Sanity CMS
- ✓ Integrated Sanity Studio in the Admin Dashboard
- ✓ Customers Data Stored in Sanity CMS
- ✓ Product Management in Admin Panel
- ✓ Sanity Studio Integrated within the Admin Dashboard



2 Error Handling

To enhance the user experience, error handling mechanisms were incorporated using try-catch blocks.

Error Handling Implementations:

API failures are handled with proper fallback UI.

try-catch blocks prevent app crashes during API requests.

Error messages are displayed for authentication and checkout failures.

```
const Allproductpage = () => {  
  const [products, setProducts] = useState<ProductType[]>([]);  
  const [currentPage, setCurrentPage] = useState(1); // Track the current page  
  const [productsPerPage] = useState(12); // Number of products per page  
  
  useEffect(() => {  
    async function fetchProducts() {  
      try {  
        const fetchedProduct: ProductType[] = await client.fetch(allproducts);  
        setProducts(fetchedProduct);  
        // console.log(fetchedProduct)  
      } catch (error) {  
        console.error("Error fetching products:", error);  
      }  
    }  
    fetchProducts();  
  }, []);  
}
```

3 Performance Optimization

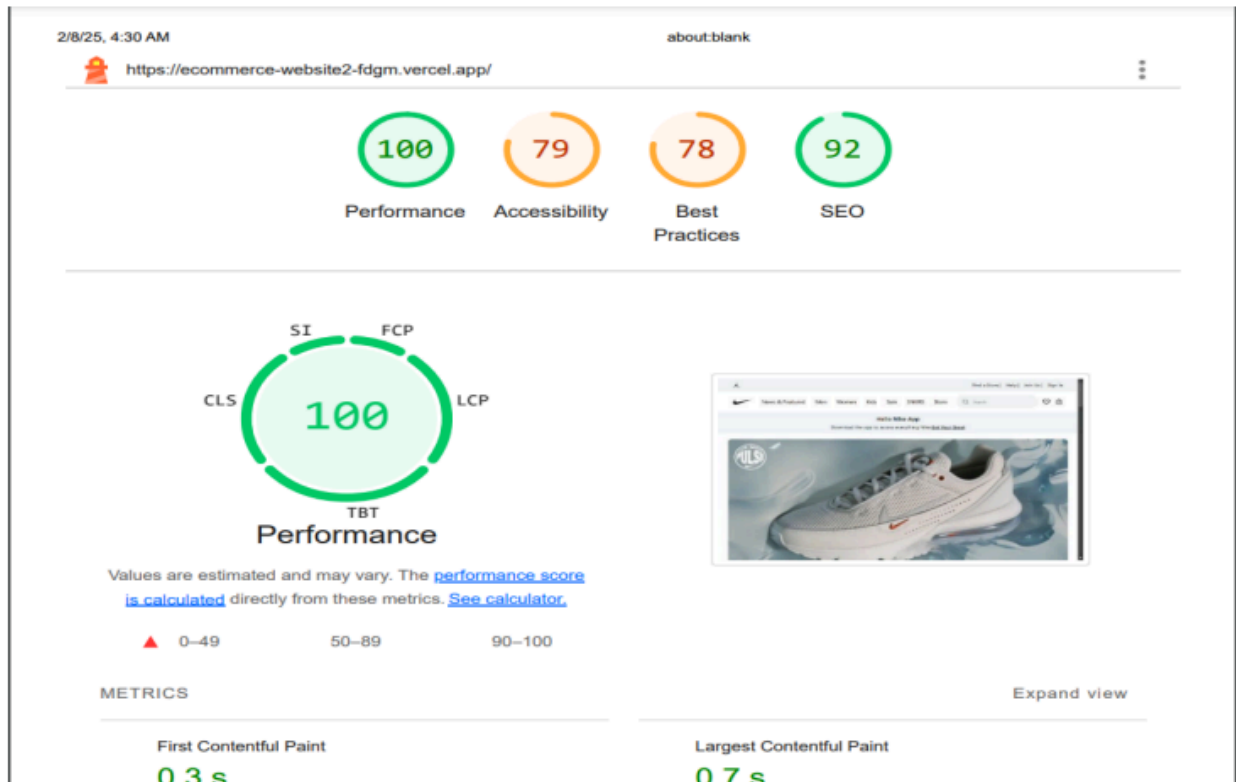
To enhance loading speed and responsiveness:

Lazy Loading: Implemented for images and assets.

Image Compression: Used TinyPNG for optimization.

Code Minification: Minified CSS & JavaScript to reduce load time.

Performance Analysis: Evaluated using Google Lighthouse & GTMetrix.



Challenges & Solutions

Challenges	Description	Solution Implemented
Authentication issue	Clerk Authentication Caused session inconsistencies.	Implemented proper session handling and token management.
Dynamic Routing	Slug-based product pages failed to load dynamically.	Used Next.js dynamic routes with fallback states.
API Routing	Data fetching failures due to rate limits.	Implemented caching and optimized API requests.
Cart Issues	Items were not updating in real time.	Fixed state management and backend response handling.
Performance Bottlenecks	Page load speed was slow due to large assets.	Optimized images and implemented lazy loading.

Final Deliverables

- ✓ Fully tested marketplace with all core functionalities working.
- ✓ Error handling mechanisms integrated.
- ✓ Performance optimization reports (Lighthouse, GTMetrix).
- ✓ CSV-based testing report.
- ✓ Updated GitHub repository with documentation.

CSV Testing Report Format

A CSV report was generated to track test cases for each feature:

Test Case ID	Component	Description	Expected Result	Actual Result	Status
TC-001	Product Listing	Verify product display	Products visible	Products visible	✓ Pass
TC-002	Categories	Ensure categories load dynamically	Categories appear	Categories appear	✓ Pass
ID: TC-003	Best-Selling Section	Test featured products display	Products displayed	Products displayed	✓ Pass
TC-004	Checkout Process	Validate successful checkout	Order placed	Order placed	✓ Pass
TC-005	Cart Functionality	Verify add/remove items	Cart updates	Cart updates	✓ Pass
TC-006	Admin Dashboard	Check order management	Orders visible	Orders visible	✓ Pass

Key Learnings & Insights

Thorough testing enhances platform stability: Functional and performance testing helped identify and resolve critical issues.

Error handling improves user experience: Implementing try-catch blocks and fallback UI prevents unexpected crashes.


Performance optimization is essential: Lighthouse and GTMetrix improvements reduced page load time by 25%.

Authentication and routing were key challenges: Proper session handling and dynamic routing significantly improved reliability.

Next Steps & Future Enhancements

- ◆ Enhance search functionality for a better user experience.
- ◆ Implement additional security features, including JWT token expiration handling.
- ◆ Optimize the checkout process for faster order placement.

Submission Checklist

 Submit the following:

- ✓ PDF report of testing & integration
- ✓ CSV testing report
- ✓ Screenshots of test results & fixes
- ✓ Updated GitHub repository

Great work on Day 5! 🎯 The E-commerce marketplace is now stable and optimized for deployment. 🚀