Day 6 - Deployment Preparation and Staging Environment – Furniro E-commerce

1. Overview:

This document outlines the steps taken to prepare for deployment and establish a staging environment for Furniro E-commerce. It covers key deployment procedures, staging environment configuration, performance assessments, testing outcomes, and final checks to ensure a seamless transition to the production environment.

2. Deployment Preparation:

2.1 Key Deployment Steps

- **Environment Configuration:**
  - Created `.env` files to securely store sensitive credentials, including API keys and URIs.
  - Managed environment-specific variables appropriately.
- **Build Optimization:**
  - Generated a production build using `next build` to enhance performance.
- **Hosting and Deployment Service:**
  - Selected Vercel for hosting due to its seamless integration with Next.js.
  - Connected the GitHub repository to enable continuous deployment upon each push to the main branch.
- **Database Connection:**
  - Configured the Sanity CMS backend for both staging and production environments with distinct datasets.
  - Verified secure and efficient database access with proper CORS settings.
- **Image Optimization:**
  - Compressed all images using Next.js `next/image` component.
  - Enabled lazy loading for non-critical images to improve initial page load time.

3. Staging Environment:

3.1 Purpose of Staging Environment

- To test the application in a live-like environment before deploying to production.
- To identify and resolve potential issues related to hosting, performance, or compatibility.

3.2 Setup

- **Domain:**
  - Deployed a staging version of the application at https://furniro-ecommerce-webclone.vercel.app/.
- **Staging Data:**

- Populated the staging database with sample data for categories, products, and user information.

4. Performance Report:

4.1 Lighthouse Performance Metrics

The following performance metrics were recorded using Google Lighthouse:

Metric	Score
Performance	99/100
Accessibility	90/100
Best Practices	100/100
SEO	100/100

5. Testing Report:

5.1 Summary of Test Cases

A total of 8 test cases were executed, covering critical functionalities such as user authentication, dynamic routing, and the checkout flow.

Test Case ID	Test Case Title	Expected Result	Actual Result	Status	Security Level	Remarks
TC-001	Search Bar Functionality	Products matching the search query are displayed.	**Pass**	Resolved	Medium	-
TC-002	Filter Component	Products are filtered correctly.	**Pass**	Resolved	Medium	-
TC-003	Pagination Component	Products are paginated correctly.	**Pass**	Resolved	Medium	-
TC-004	Checkout Validation	User is prompted to fill required fields.	**Pass**	Resolved	High	-
TC-005	Dynamic Routing	Dynamic product page loads correctly.	**Pass**	Resolved	High	-

6. Security Measures:

1. Input Validation:
   - All user inputs are sanitized to prevent SQL injection or XSS attacks.
2. API Security:
   - Secured API endpoints.
   - Ensured API keys and secrets are stored in environment variables and never exposed in the frontend.

3. HTTPS:
    - Enabled HTTPS for secure communication between the client and server.
4. CORS Configuration:
    - Configured CORS policies to allow only trusted domains to access backend APIs.

7. Final Checklist for Deployment:

- Verified production build with **next build**.
- Tested all critical functionalities in the staging environment.
- Ensured all sensitive information is securely stored.
- Configured proper error monitoring and analytics.
- Deployed staging build to **Vercel** for final review.