

DAY 6- DEPLOYMENT PREPARATION AND STAGING ENVIRONMENT SETUP

The purpose of Day 6 is to prepare the Bandage Marketplace for deployment by setting up a staging environment, configuring hosting platforms, and ensuring readiness for a public-facing application. The primary focus is to create a pre-production environment for rigorous testing and seamless operation. It also includes gaining familiarity with managing environments like non-production (TRN, DEV, SIT) and production (UAT, PROD, DR), adhering to industry deployment standards.

Learning Objectives

1. Develop interactive frontend modules that dynamically retrieve data from Sanity CMS or external APIs.
 2. Design reusable, scalable components for easy updates and maintenance.
 3. Apply effective state management techniques for smooth data flow across modules.
 4. Enhance user experience with responsive designs and UX/UI best practices.
 5. Simulate professional workflows to prepare for real-world projects.
-

Types of Professional Environments

1. **Training (TRN):**
Purpose: Used to train team members and for practice.
Key Feature: Allows users to explore the system without affecting active setups.
2. **Development (DEV):**
Purpose: Dedicated space for developers to code and perform local testing.
Key Feature: Enables iterative coding and debugging without impacting live systems.
3. **System Integration Testing (SIT):**
Purpose: Verifies the communication between various components and systems.
Key Feature: Ensures seamless interaction and compatibility of subsystems.
4. **User Acceptance Testing (UAT):**
Purpose: Allows users to validate application functionality against business goals.
Key Feature: Ensures alignment with user expectations for deployment readiness.
5. **Production (PROD):**
Purpose: The final, live platform for end users.
Key Feature: Guarantees optimal performance, security, and availability for real-world use.

6. Disaster Recovery (DR):

Purpose: Backup system for emergencies like failures or disasters.

Key Feature: Facilitates quick recovery to reduce downtime in critical situations.

Key Areas of Focus

1. Planning Deployment Strategies

- Selected **Vercel** as the deployment platform for staging and production.
- Integrated Sanity CMS for dynamic content using secure API tokens and dataset IDs.

2. Setting Up Environment Variables

- Stored sensitive information like API keys and tokens in a `.env.local` file.
- Configured these variables securely within the Vercel Dashboard for deployment.

3. Staging Setup

- Successfully deployed the application to the Vercel platform.
- Validated data retrieval from Sanity CMS in the staging environment.

4. Testing in Staging

- Conducted functional tests using Cypress, API validation with Postman, and performance analysis using Lighthouse.
- Verified HTTPS implementation, responsive design, and proper handling of sensitive data.

5. Updating Documentation

- Compiled deployment guides, configurations, and testing outcomes in a `README.md` file.
- Added all supporting files and reports to the GitHub repository.

Steps for Implementation:

Step 1: Hosting Platform Setup

• Choose a Platform:

- You can choose a platform like **Vercel** or **Netlify** for simple, quick deployment if you need something hassle-free.
- If you require more advanced configurations, greater control, or scalable infrastructure, platforms like **AWS** or **Azure** may be better suited to your needs.

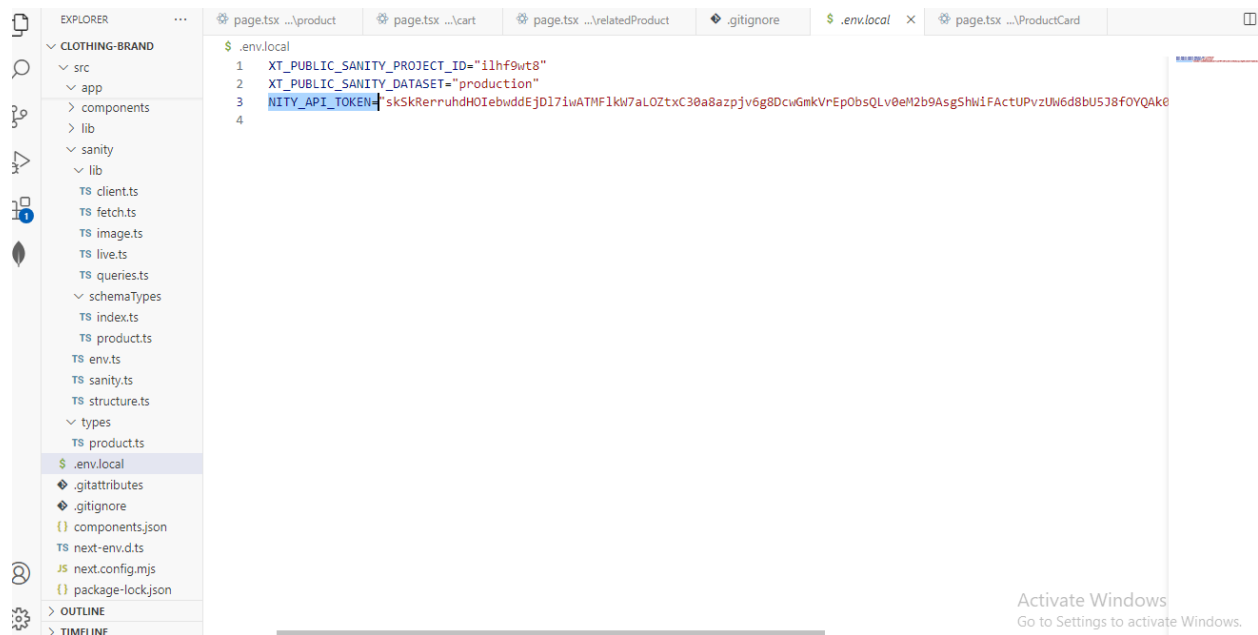
• Connect Repository:

- Link your **GitHub** repository to the hosting platform for seamless integration.
- Be sure to configure the **build settings** and ensure necessary **deployment scripts** are added to ensure smooth deployment (e.g., for Node.js apps, ensure you have the build command like `npm run build` and the correct output directory set).

Step 2: Configure Environment Variables

Create a .env File:

- In your project's root directory, create a `.env` file to securely store **environment variables**. This might include sensitive information like API keys, database URLs, and other project-specific configurations.
- For example



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a `.env.local` file highlighted. The code editor shows the contents of `.env.local`:

```
1 XT_PUBLIC_SANITY_PROJECT_ID="ilhf9wt8"  
2 XT_PUBLIC_SANITY_DATASET="production"  
3 NITY_API_TOKEN="skSkRerruhdHOIebwddEjDl7IwATHF1kh7aLOZtxC30a8azpjv6g8DcwGmkVrEpObsQLv0eM2b9AsgShw1FactUPvzUw6d8bU5J8fOYQAK8"  
4
```

Set Environment Variables on Hosting Platform:

- Most hosting platforms (Vercel, Netlify, AWS, etc.) provide an interface to add environment variables directly within their settings.
 - Go to the platform's dashboard and locate the **Environment Variables** section under the project settings. Add the variables you defined in the `.env` file to these settings for production.
- **Test Locally:**
 - Before triggering deployment, make sure everything works locally, including the environment variables. Run the project locally with commands like `npm run dev` or equivalent and ensure that all functionalities are operational.

- **Deploy:**

- Once tested locally, trigger the deployment process. Most platforms will allow you to deploy directly from GitHub, or you can configure deployment via your CI/CD tool.
- After deployment, check your live app to ensure it works as expected. Monitor deployment logs for any errors that might arise.

Step 4: Staging Environment Testing

Testing Types:

o Functional Testing:

- Verify key features like **product listing**, **search**, and **cart operations** to ensure they work as expected.

o Performance Testing:

- Use tools like **Lighthouse** or **GTmetrix** to analyze **page load speed**, **responsiveness**, and **performance metrics** such as **FCP**, **TTI**, and **LCP**.

o Security Testing:

- Ensure **input fields** are secure from attacks, all communications use **HTTPS**, and **API calls** are properly encrypted and validated.

2. Test Case Reporting:

Test Case ID	Test Case Description	Test Steps	Expected Result	Actual Result	Status	Severity Level	Remarks
TC001	Product Display	Verify that all items are shown on the homepage.	All products should be visible correctly.	All products displayed as intended.	Pass	High	No issues encountered.
TC002	Product Information	Select a product to view its details.	The product details page should load without issue.	Product details page loaded successfully.	Pass	High	No issues encountered.
TC003	Add to Cart	Click the 'Add to Cart' button for a product.	The product should be successfully added to the cart.	Product added to the cart without any issues.	Pass	High	No issues encountered.
TC004	Cart Management	Add and remove products from the cart.	The cart should reflect the correct number of items after addition or removal.	Cart updated correctly when items were added or removed.	Pass	High	No issues encountered.
TC005	Navigation	Select a product to go to its detail page.	The relevant product detail page should open.	Correct product page opened with accurate details.	Pass	Medium	No issues encountered.
TC006	Category Filtering	Apply various category filters.	The displayed products should match the selected filter.	Products filtered accurately based on the chosen category.	Pass	Medium	No issues encountered.
TC007	Error Handling (Network)	Simulate a network failure and attempt to load a product.	An error message should notify the user about the failure.	Appropriate error message displayed as expected.	Pass	Critical	Error handling functions correctly.
TC008	Error Handling (Invalid Data)	Input incorrect data into the search bar or form.	An error message should alert the user.	Error message shown for invalid input.	Pass	High	Invalid input handled with a clear message.
TC009	Mobile Compatibility	Test the website on various devices (desktop, tablet, mobile).	The website should be responsive and adjust its layout on all devices.	The website's design adapts to different screen sizes effectively.	Pass	Medium	No issues encountered on mobile/tablet.

3. Performance Testing:

Below is a performance report generated by tools like Lighthouse in your GT Matrix.

Get full report details and more testing capacity with a FREE GTmetrix account.

Log in

Create a FREE account

1235 555-0118

1235 555-0118

Follow Us

Bandage

Home Shop About Blog Contact Pages

NEW COLLECTION

SHOP NOW

NEW COLLECTION

SHOP NOW

Latest Performance Report for:

https://clothing-brand-liart.vercel.app/

Report generated: Mon, Jan 27, 2025 7:42 PM -0800

Test Server Location: Vancouver, Canada

Using: Chrome 117.0.0.0, Lighthouse 11.0.0

GTmetrix Grade ?

Web Vitals ?

A

Performance ?

100%

Structure ?

92%

LCP ?

386ms

TBT ?

0ms

CLS ?

0

Activate Wi

Is it Resilient?

✓ Looks great! This site had no render-blocking 3rd party requests that could be a single point of failure. It had no security issues. HTML content was mostly generated server-side.

Opportunities 0 Tips 0 P10 Experiments 0

You have Free Experiments Available! [Try them now!](#)

Page Performance Metrics

(Based on Median Run by: [Speed Index](#))

Note: Metric availability will vary

First View (Run 2)

Time to First Byte	Start Render	First Contentful Paint	Speed Index	Largest Contentful Paint	Cumulative Layout Shift	Total Blocking Time	Page Weight
.714s	2.082s	2.065s	2.489s	3.056s	0	.382s	1,298KB
When did the content start downloading?	When did pixels first start to appear?	How soon did text and images start to appear?	How soon did the page look usable?	When did the largest visible content finish loading?	How much did the design shift while loading?	Was the main thread blocked?	How many bytes downloaded?

Visual Page Loading Process ([Explore](#))

0.0s	0.1s	0.2s	0.3s	0.4s	0.5s	0.6s	0.7s	0.8s	0.9s	1.0s	1.1s	1.2s	1.3s	1.4s	1.5s	1.6s	1.7s	1.8s	1.9s	2.0s	2.1s	2.2s	2.3s	2.4s

Go to Settings to activate Windows.