DAY 5 - TESTING, ERROR HANDLING, AND BACKEND INTEGRATION REFINEMENT

1. Objective

The objective for Day 5 is to prepare the marketplace for real-world deployment by focusing on comprehensive testing, optimization, and refinement of the user experience. This involves ensuring all features and functionalities are operating smoothly, securing the platform against vulnerabilities, and enhancing its performance for a seamless launch. Achieving this readiness requires meticulous attention to testing protocols, error handling mechanisms, and cross-platform compatibility.

2. Key Learning Objectives

The goal of Day 5 is to equip you with the necessary skills to prepare a marketplace for deployment. Key areas of focus include:

- **Comprehensive Testing Mastery**: Understanding functional, non-functional, user acceptance, and security testing to ensure all components perform optimally.
- **Error Management Skills**: Learning how to implement effective error-handling mechanisms that enhance user trust and satisfaction.
- **Performance Optimization**: Developing techniques to optimize platform speed, responsiveness, and scalability, ensuring it can handle real-world usage.
- **Cross-Device Compatibility**: Gaining expertise in making the marketplace adaptable and accessible across various devices and browsers.
- **Documentation Proficiency**: Mastering the creation of detailed testing documentation for transparency, maintenance, and troubleshooting purposes.

3. Key Learning Outcomes

By the end of Day 5, the following outcomes should be achieved:

- **Tested and Secure Platform**: A thoroughly tested marketplace with all functionalities verified, vulnerabilities addressed, and user data secured.
- **Enhanced User Experience**: A marketplace with efficient error-handling mechanisms and optimized performance, ensuring a smooth and frustration-free user experience.

- **Scalability and Speed**: A platform capable of handling high traffic with optimized speed, achieved through techniques like lazy loading, code splitting, and caching strategies.
- **Universal Accessibility**: Compatibility across all major browsers and devices, ensuring users can access the marketplace regardless of their chosen platform.
- **Detailed Documentation**: Comprehensive records of testing processes, results, and resolutions, enabling efficient future troubleshooting and platform maintenance.

Step 1: Functional Testing and Key Features to Test

Functional testing ensures that the marketplace operates as intended by verifying each feature against defined requirements. This step involves simulating user actions to identify and fix issues before deployment.

Key Features to Test:

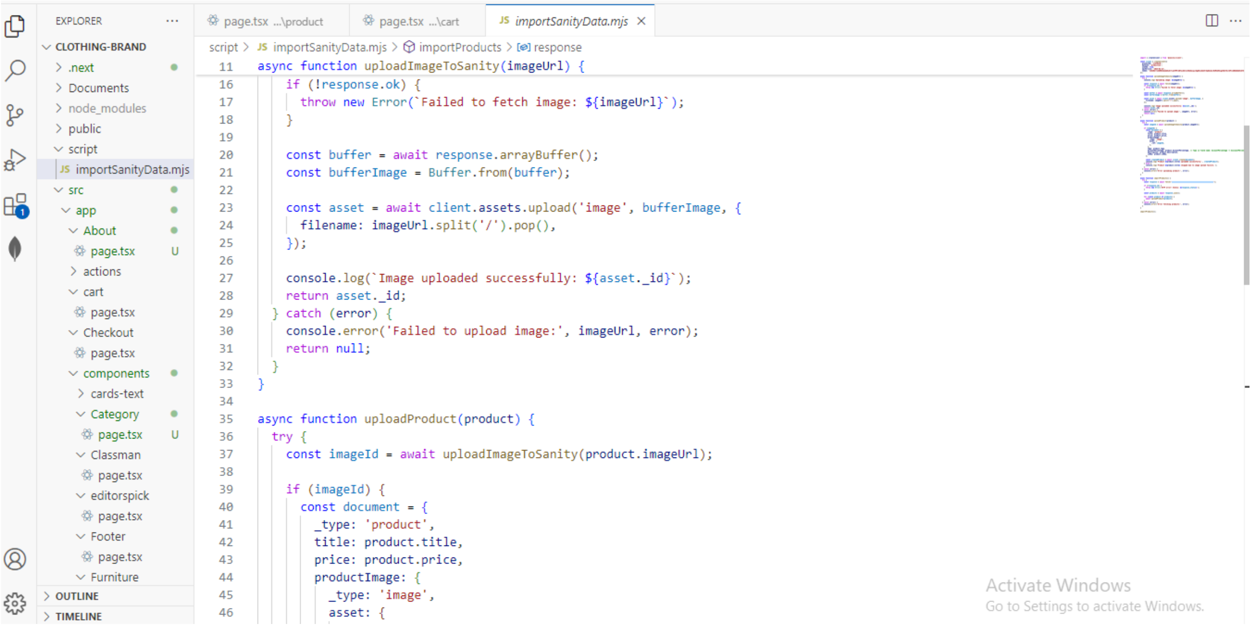
1. **User Registration and Login**: Validate account creation, login, and password recovery processes to ensure secure and seamless access.
2. **Search and Navigation**: Test search accuracy, filtering, and site navigation for an intuitive user experience.
3. **Product Listing and Management**: Ensure accurate display and management of product details, stock updates, and edits.
4. **Shopping Cart and Checkout**: Verify adding/removing items, calculating totals, and processing payments securely.
5. **Payment and Refunds**: Test multiple payment methods, failed transactions, and refund handling.
6. **User Account Management**: Confirm users can update profiles, manage preferences, and access order history securely.
7. **Admin Dashboard**: Validate tools for monitoring sales, managing roles, and moderating content.

Step 2: Error Handling

Effective error handling ensures a smooth user experience by managing issues effectively and minimizing disruptions.

Key Elements:

1. **User-Friendly Messages**: Provide clear, actionable error messages (e.g., “An error occurred. Please try again.”).
2. **Fallback Mechanisms**: Use spinners, save progress, or provide alternate options during errors.
3. **Error Logging**: Log detailed error data for developers to resolve issues quickly.
4. **API Error Management**: Handle API failures with retries and user-friendly updates.
5. **Testing**: Simulate common errors to ensure they are handled properly.



```javascript
async function uploadImageToSanity(imageUrl) {
    if (!response.ok) {
      throw new Error(`Failed to fetch image: ${imageUrl}`);
    }

    const buffer = await response.arrayBuffer();
    const bufferImage = Buffer.from(buffer);

    const asset = await client.assets.upload('image', bufferImage, {
      filename: imageUrl.split('/').pop(),
    });

    console.log(`Image uploaded successfully: ${asset._id}`);
    return asset._id;
  } catch (error) {
    console.error('Failed to upload image:', imageUrl, error);
    return null;
  }
}

async function uploadProduct(product) {
  try {
    const imageId = await uploadImageToSanity(product.imageUrl);

    if (imageId) {
      const document = {
        _type: 'product',
        title: product.title,
        price: product.price,
        productImage: {
          _type: 'image',
          asset: {
```

Step 3: Performance Optimization

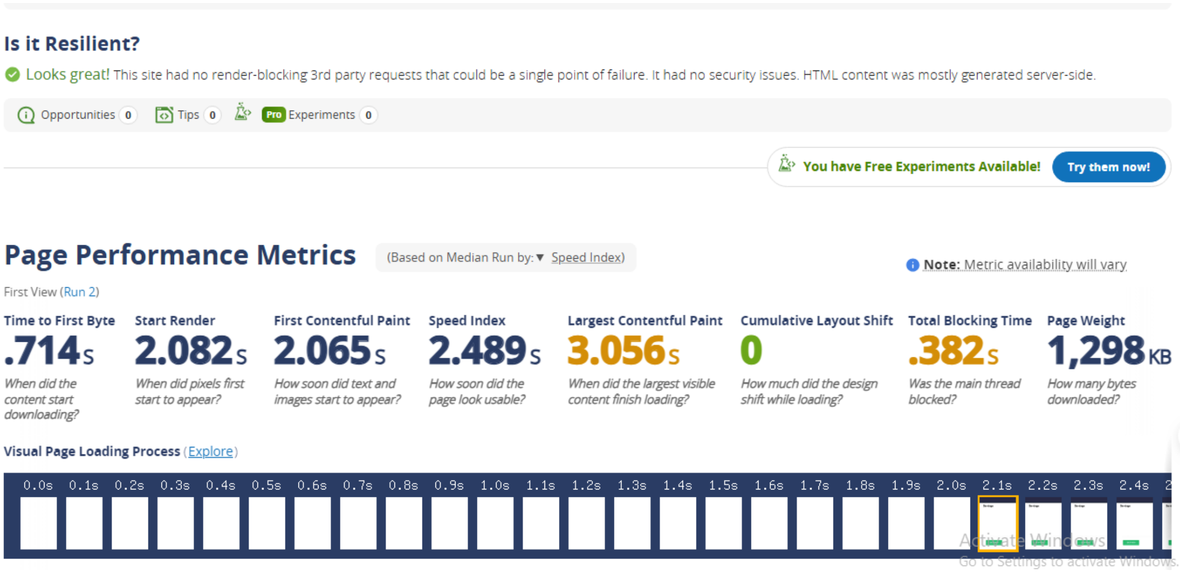
Performance optimization ensures the marketplace is fast and responsive, even under heavy traffic.

Key Strategies:

- **Lazy Loading**: Load images and content only when needed.
- **Caching**: Use browser and server-side caching to reduce loading times.
- **Code Optimization**: Minimize and bundle JavaScript and CSS files.
- **CDN Utilization**: Serve static assets through a Content Delivery Network for faster delivery.
- **Database Tuning**: Optimize queries to handle large-scale operations efficiently.
- **WebPageTest & GTmetrix**: Use these tools to identify performance bottlenecks and improve page load speeds.

Outcome:

A seamless user experience with faster load times and the ability to handle peak traffic.



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.

Page Performance Metrics (Based on Median Run by: ▾ Speed Index)

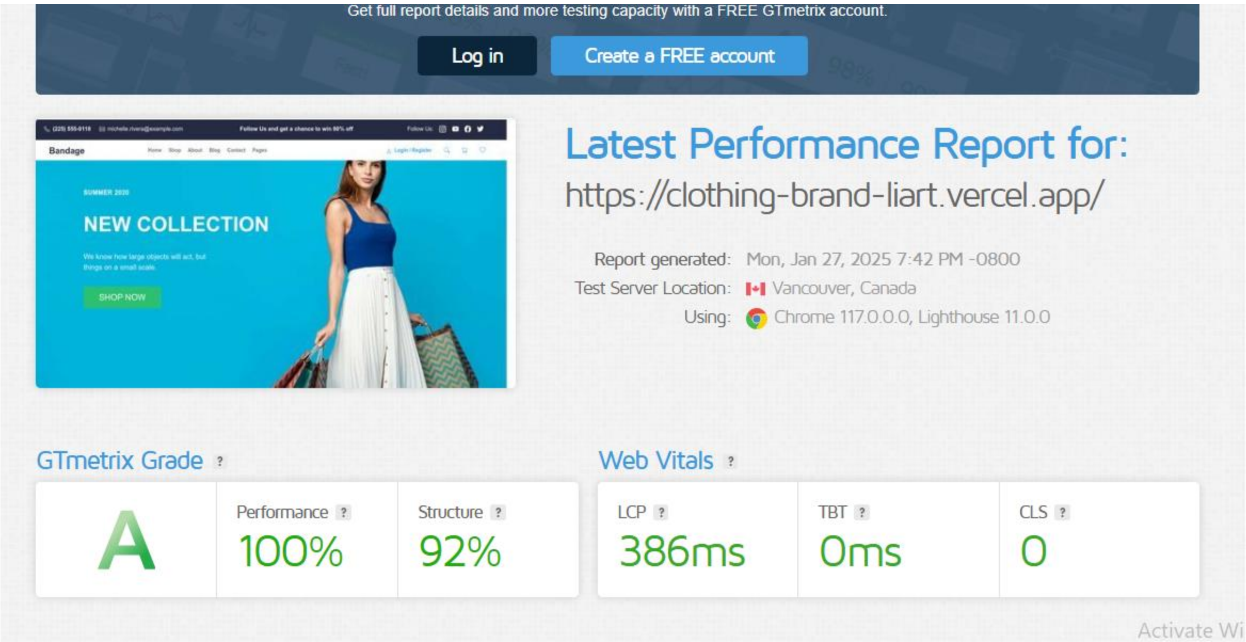
Note: Metric availability will vary

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



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

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 ?

	Performance ?	Structure ?
A	100%	92%

Web Vitals ?

LCP ?	TBT ?	CLS ?
386ms	0ms	0

Step 4: Cross-Browser and Device Testing

Cross-browser and device testing ensures the marketplace works across all platforms.

Key Activities:

- Test on major browsers (Chrome, Firefox, Safari, Edge) and various devices (desktop, tablets, smartphones).
- Use tools like BrowserStack for automated compatibility checks.
- Validate responsive design for smooth navigation on different screen sizes.

Outcome:
A consistent user experience regardless of browser or device.

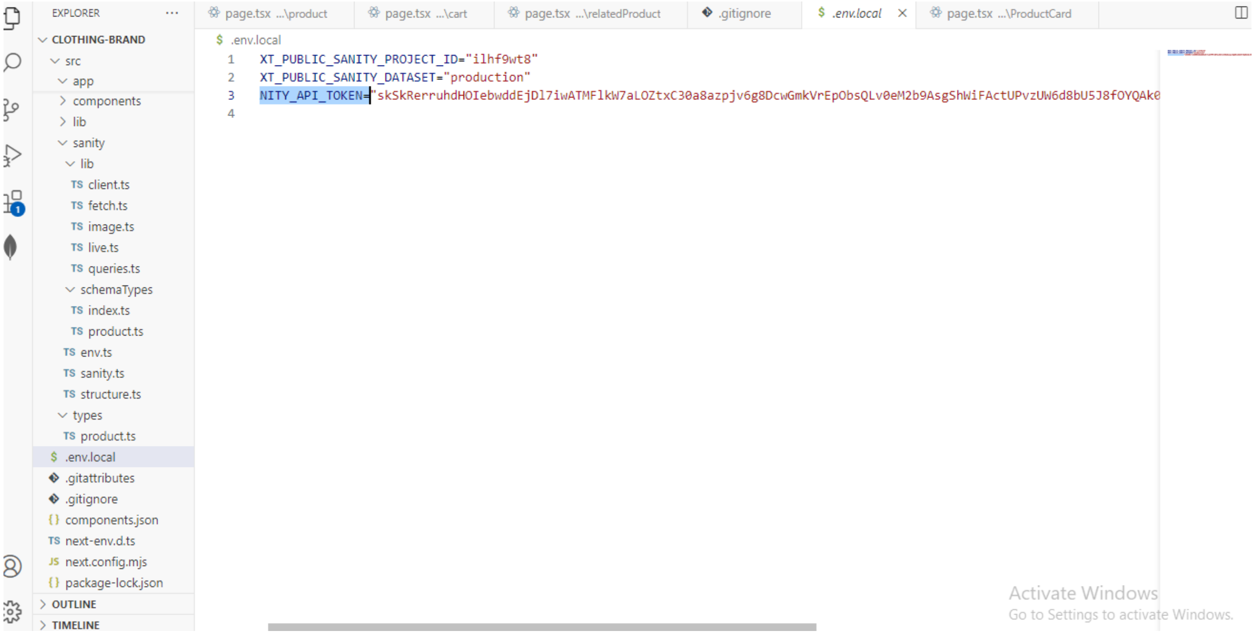
Step 5: Security Testing

Security testing safeguards user data and prevents vulnerabilities.

Key Activities:

- Perform vulnerability assessments to identify weak points.
- Test for SQL injection, cross-site scripting (XSS), and data leaks.
- Ensure secure data transmission through encryption protocols (e.g., HTTPS).
- Authentication & Authorization: Implement secure user authentication mechanisms like JWT (JSON Web Tokens) and OAuth. Ensure proper authorization checks are in place to restrict access to certain features or resources.

Outcome:
A secure marketplace that protects user data and maintains trust.



.env.local

```
XT_PUBLIC_SANITY_PROJECT_ID="ilhf9wt8"
XT_PUBLIC_SANITY_DATASET="production"
NITY_API_TOKEN="skSkRerruhdHOIebwddEjDl7iwATMFlkW7aLOZtxC30a8azpjv6g8DcwGmkVrEpObsQLv0eM2b9AsgShWiFActUPvzUW6d8bU5J8fOYQAk0

```

Step 6: User Acceptance Testing (UAT)

User Acceptance Testing (UAT) ensures the platform meets business needs and user expectations.

Key Activities:

- Involve stakeholders and end-users to test workflows.
- Verify that all features function as required in real-world scenarios.
- Collect feedback for final adjustments.

Outcome:
A user-approved platform ready for deployment.

Step 7: Documentation Updates

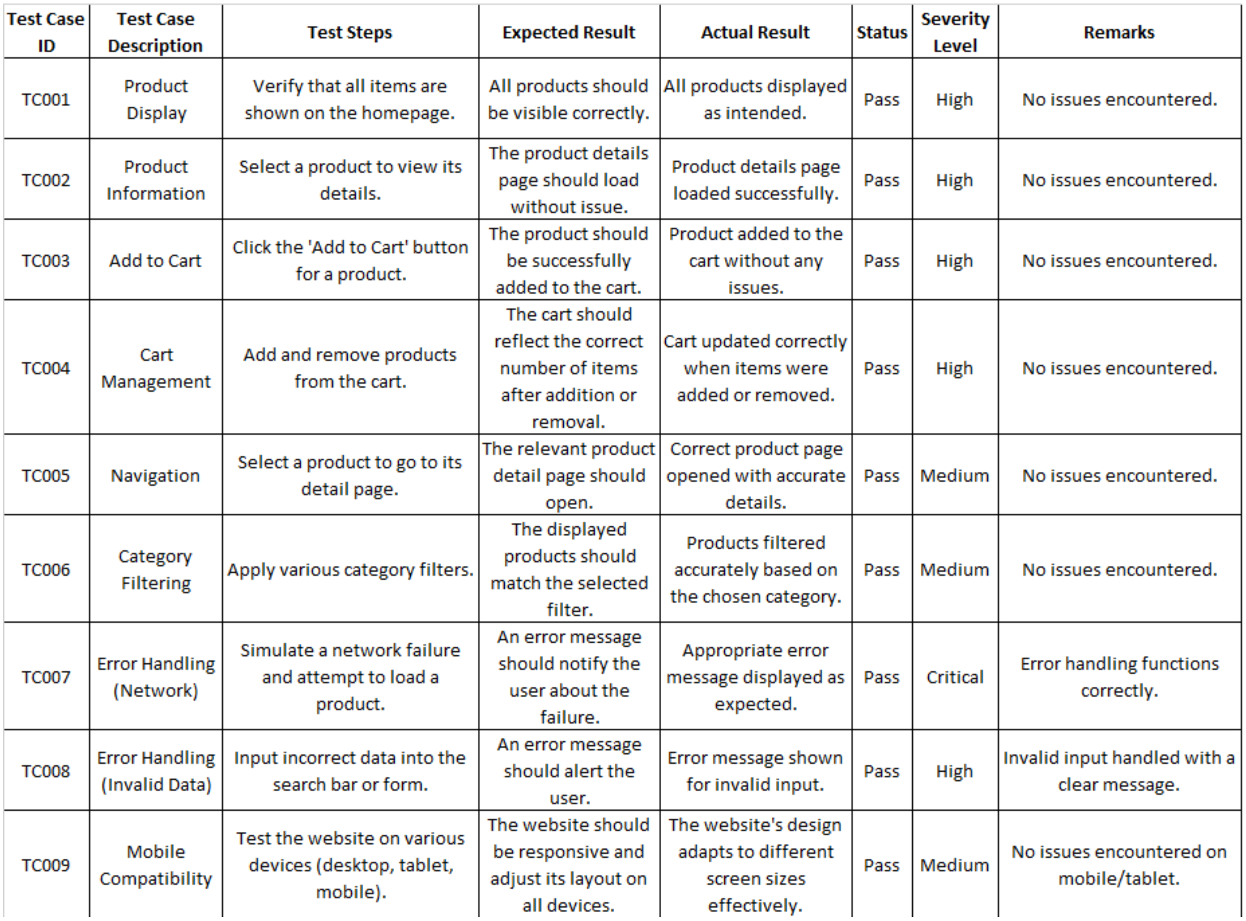
Comprehensive documentation ensures transparency and supports future maintenance.

Key Updates:

- Document test cases, results, and performance benchmarks.
- Maintain a detailed bug report and resolution log.
- Update user manuals and FAQs for end-users.

Outcome:
Clear, updated documentation to guide developers, testers, and users.

I Have followed these steps, and ensure the marketplace is thoroughly tested, optimized, and ready for deployment. With effective error handling, robust performance optimizations, and cross-device compatibility, the platform will offer a seamless user experience, making it both secure and scalable for real-world use.



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.